



## Systems Reference Library

### **IBM 1410/7010 Operating System (1410-PR-155) COBOL**

This publication is designed to be used by programmers in conjunction with the publication, *IBM General Information Manual, COBOL*, Form F28-8053, and contains additional specifications required to write COBOL programs to be processed under the 1410/7010 Operating System.

The similarity between COBOL and ordinary business English provides programmers with a convenient method for writing source programs. Source program statements are translated directly into machine language by the COBOL compiler (1410-CB-969), which takes full advantage of the capabilities of the IBM 1410 and 7010 Data Processing Systems.

MAJOR REVISION (November 1963)

This publication supersedes the preliminary reference manual,  
*IBM 1410/7010 Operating System; COBOL*, Form C28-0327,  
with its associated Technical Newsletter (N28-1080).

Copies of this and other IBM publications can be obtained through IBM Branch Offices.  
Address comments concerning the contents of this publication to:  
IBM Corporation, Programming Systems Publications, Dept. D91, PO Box 390, Poughkeepsie, N. Y.

## Contents

<b>Introduction</b> .....	5	ADD .....	21
Purpose of this Publication .....	5	SUBTRACT .....	22
Purpose of the Language .....	5	MULTIPLY .....	22
Prerequisite and Related Information .....	5	DIVIDE .....	22
Machine Requirements .....	5	COMPUTE .....	23
Acknowledgment .....	5	Procedure Branching Verbs .....	23
COBOL Language Forms and Notations .....	6	GO TO .....	23
Identification Division .....	7	ALTER .....	23
 <b>Environment Division</b> .....	8	PERFORM .....	23
Structure of the Environment Division .....	8	Compiler Directing Verbs .....	23
Configuration Section .....	8	ENTER .....	23
SOURCE-COMPUTER Paragraph .....	8	EXIT .....	24
OBJECT-COMPUTER Paragraph .....	8	NOTE .....	24
SPECIAL-NAMES Paragraph .....	8	Ending Verb .....	24
Input-Output Section .....	9	STOP .....	24
FILE-CONTROL Paragraph .....	9	Conditional Expressions .....	24
I-O-CONTROL Paragraph .....	10	Added Features of the Procedure Division .....	24
 <b>Data Division</b> .....	11	 <b>General Information</b> .....	25
IBM 1410/7010 Files and Records .....	11	Programming Techniques .....	25
Recording Modes .....	11	Compatibility Considerations .....	25
Standard Tape Labels .....	11	Qualification of Names .....	26
Record Formats for Tape Files .....	12	Literals .....	26
Record Formats for Unit-Record Files .....	12	Character Sets .....	26
File Section .....	13	Figurative Constants .....	26
File Description Entry .....	13	TALLY .....	27
Record Description Entry .....	14	MONITOR-DATE .....	27
Working-Storage and Constant Sections .....	16	Class Conditions .....	27
Added Features of the Data Division .....	16	1410/7010 COBOL Compiler Requirements .....	28
 <b>Procedure Division</b> .....	18	Requirements for Compilation .....	28
Compiler Directing Declaratives .....	18	EXEQ Card Operand Options .....	28
USE Verb .....	18	Requirements for Execution .....	28
Input/Output Verbs .....	19	The Subprogram TITLE Card .....	28
OPEN and CLOSE .....	19	IDENT Field of PROGRAM-ID Card .....	28
READ .....	19	Multiple Subprogram COBOL Output .....	29
WRITE .....	19	Control Card Requirements .....	29
DISPLAY .....	20	 <b>Appendixes</b> .....	31
ACCEPT .....	20	A: COBOL Words .....	31
Data Manipulation Verbs .....	20	B: Organization of Source Program .....	32
MOVE .....	20	C: Object Time Error Analysis and Messages .....	33
EXAMINE .....	21	D: Diagnostic Messages .....	34
Arithmetic Verbs .....	21	E: Sample Problem .....	36
		 <b>Index</b> .....	40



### Purpose of this Publication

This publication is designed to be used by programmers in conjunction with the publication, *IBM General Information Manual, COBOL*, Form F28-8053, and contains additional specifications required to write COBOL programs to be processed under the 1410/7010 Operating System.

### Purpose of the Language

The similarity between COBOL and ordinary business English provides programmers with a convenient method for writing source programs. Source program statements are translated directly into machine language by the COBOL compiler, which takes full advantage of the capabilities of the IBM 1410 and 7010 Data Processing Systems.

### Prerequisite and Related Information

A basic knowledge of COBOL and the IBM 1410/7010 Operating System is required, and a knowledge of the IBM 1410 or 7010 Data Processing System is recommended in order to fully understand the information presented in this publication.

Anyone without this prior knowledge is requested to read the following publications:

*IBM General Information Manual, COBOL*, Form F28-8053

*IBM 1410/7010 Operating System; Basic Concepts*, Form C28-0318

*IBM 1410 Principles of Operation*, Form A22-0576 or  
*IBM 7010 Principles of Operation*, Form A22-6726

The following IBM 1410/7010 Operating System publications are mentioned in this manual and should be available for reference purposes:

*System Monitor*, Form C28-0319

*System Generation*, Form C28-0352

*Basic Input/Output Control System*, Form C28-0322  
*Operator's Guide*, Form C28-0351

In order to make full use of the 1410/7010 COBOL language, and to run COBOL programs within the framework of the Operating System, the reader must be familiar with the contents of the publication, *System Monitor*.

### Machine Requirements

The minimum machine requirements for compiling programs using the COBOL compiler are included in the publication, *System Generation*, Form C28-0352. However, machine requirements for running an object program depend upon the nature of the program.

### Acknowledgment

In accordance with the requirements of the official government manual, *COBOL-1961-Extended*, Form number 1962-0668996, describing COBOL (obtained by sending a purchase order and \$1.25 to: Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C.), the following extract from that manual is presented for the information and guidance of the user:

"This publication is based on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force  
Bureau of Standards, United States Department of Commerce  
Burroughs Corporation  
David Taylor Model Basin, Bureau of Ships, United States Navy  
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company  
International Business Machines Corporation  
Radio Corporation of America  
Sylvania Electric Products, Inc.  
UNIVAC Division of Sperry Rand Corporation

"In addition to the organizations listed above, the following other organizations participated in the work of the Maintenance Group:

Allstate Insurance Company  
The Bendix Corporation, Computer Division  
Control Data Corporation  
E. I. du Pont de Nemours and Company  
General Electric Company  
General Motors Corporation  
Lockheed Aircraft Corporation  
The National Cash Register Company  
Philco Corporation

Royal McBee Corporation  
Standard Oil Company (New Jersey)  
United States Steel Corporation

"This COBOL-61 manual is the result of contributions made by all of the above-mentioned organizations. No warranty, expressed or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

"It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

"Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures and the methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

"The authors and copyright holders of the copyrighted material used herein: FLOW-MATIC\*. Programming for the UNIVAC\* I and II, Data Automation Systems © 1958, 1959, Sperry Rand Corporation; IBM Commercial Translator, Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell, have specifically authorized the use of this material, in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

"Any organization interested in reproducing the COBOL report and initial specifications, in whole or in part, using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention 'COBOL' in acknowledgment of the source, but need not quote this entire section."

\*Trademark of Sperry Rand Corporation

## COBOL Language Forms and Notations

Throughout this publication, all the basic forms are prescribed for the various verbs, clauses, entries, and other essential elements of the 1410/7010 COBOL language. These are generalized forms intended to guide the programmer in writing his own statements. If statements are written in formats other than those presented in this manual, the compilation will result in error.

The following rules of notation have been followed in the presentation of these forms:

1. All words printed entirely in capital letters are COBOL words; i.e., words that have preassigned meanings in the COBOL system.

2. All underlined words are required unless the portion of the format containing them is itself optional; i.e., enclosed in square brackets. These are key words and if any such word is missing or is incorrectly spelled, it is an error in the program.

3. All COBOL words not underlined may be included or omitted at the option of the programmer. These words are optional and are used only for the sake of readability. Misspelling, however, constitutes an error.

4. All italicized words represent information that must be supplied by the programmer. The nature of the information required is indicated in each case. In most instances, the programmer will be required to provide an appropriate data-name, procedure-name, literal, etc.

5. Material enclosed in square brackets [ ] may be used or omitted as required by the programmer.

6. When material is enclosed in braces { }, only one of the enclosed items is required; the others are to be omitted. The choice is to be made by the programmer.

7. Punctuation, where shown, is essential. Other punctuation may be inserted by the programmer in accordance with the rules specified in the General Information Manual.

8. In certain cases, a succession of operands or other elements may be used in the same statement. In such a case, this possibility is indicated by the use of three dots following the item affected. The dots apply to the last complete element preceding them; thus, if a group of operands and key words are enclosed within brackets, and three dots precede the closing bracket, the entire group must be repeated if any repetition is required, not merely the last operand.

## Identification Division

The information specified in the Identification Division of the source program allows the programmer to identify or label his program, and provide other pertinent information concerning the program. This division must precede the other divisions when the source program is presented to the compiler. The over-all structure of the Identification Division is:

IDENTIFICATION DIVISION.

PROGRAM-ID. *program-name.*

[ AUTHOR. *author-name.* ]

[ INSTALLATION. *any sentence or group of sentences.* ]

[ DATE-WRITTEN. *any sentence or group of sentences.* ]

[ DATE-COMPILED. *any sentence or group of sentences.* ]

[ SECURITY. *any sentence or group of sentences.* ]

[ REMARKS. *any sentence or group of sentences.* ]

Usage of the IDENT\* portion of the PROGRAM-ID source statement is explained in the section, "1410/7010 COBOL Compiler Requirements."

For additional details concerning the Identification Division, see the General Information Manual.

\*Columns 73-80 of the *COBOL Program Sheet* (Reference Format)

## Environment Division

In this part of the COBOL source program, the programmer describes to the compiler the physical characteristics of the IBM 1410 or 7010 System that will be used to compile the source program, and the system that will be used to execute the object program. This division must immediately follow the Identification Division when the source program is submitted to the compiler.

### Structure of the Environment Division

The over-all structure of the Environment Division for a source program is given below for reference purposes:

```
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER. — — — .  
OBJECT-COMPUTER. — — — .  
SPECIAL-NAMES. — — — .  
INPUT-OUTPUT SECTION.  
FILE-CONTROL. — — — .  
I-O-CONTROL. — — — .
```

Each of the subdivisions of the Environment Division is discussed in the following pages. This discussion is in terms of the IBM 1410/7010 COBOL compiler, and therefore includes specifications not contained in the General Information Manual.

### Configuration Section

The three paragraphs of the Configuration Section specify, respectively, the computer on which the COBOL compiler is to be run, the computer on which the object program is to be run, and the names of the machine devices and switch conditions referred to by the programmer in the Procedure Division of his source program.

#### SOURCE-COMPUTER Paragraph

The purpose of this paragraph is to specify the computer on which the COBOL compiler is to run to compile the source program. The general form of this paragraph for the 1410/7010 COBOL compiler is:

SOURCE-COMPUTER. { IBM-1410 }  
                          { IBM-7010 }

Additional information regarding the source computer (e.g., actual core-storage size, core storage available, etc.) is contained in the Resident Monitor's Communication Region; therefore, no additional entries are permitted in this paragraph. (Sec *System Monitor*.)

#### OBJECT-COMPUTER Paragraph

The purpose of this paragraph is to specify the computer on which the object program is to be executed. The general form of this paragraph is:

OBJECT-COMPUTER. { IBM-1410 }  
                          { IBM-7010 }

#### SPECIAL-NAMES Paragraph

This optional paragraph equates mnemonic-names with device-names representing certain system units or the console printer, and equates condition-names with the status of the system's Standard Input Unit end-of-file switch and/or a switch in the Resident Monitor's Communication Region. The general form of this paragraph is:

SPECIAL-NAMES. [ device-name-1 IS mnemonic-name-1  
                          [ device-name-2 IS mnemonic-name-2 ... ] ].

[ MONITOR-SWITCH  
          [ literal-1 STATUS IS condition-name-1 ]  
          [ literal-2 STATUS IS condition-name-2 ... ] ]

[ I-O-SWITCH EOF-SIU  
          [ ON STATUS IS condition-name-3 ]  
          [ OFF STATUS IS condition-name-4 ] ].

#### DEVICE-NAMES

The device-names of the SPECIAL-NAMES paragraph must be chosen from the following list:

DEVICE-NAME	DESCRIPTION
CONSOLE-PRINTER	the console printer
SYSTEM-OUTPUT-PUNCH	the Standard Punch Unit
SYSTEM-OUTPUT-PRINTER	the Standard Print Unit

System units are discussed in the publication, *System Monitor*.

#### MONITOR-SWITCH

The MONITOR-SWITCH is used to represent a single-character switch position within the Resident Monitor's Communication Region, which is set by the operator with the \$3x console inquiry. This switch can be referred to in the Procedure Division by means of a con-





DEVICE-NAME	DESCRIPTION
CARD-READER xxx	is the standard device-name for the card reader of the 1402 Card Read Punch, or the 1442 Card Reader. "xxx" is the name of a symbolic unit (e.g., MR1).*
CARD-PUNCH xxx	is the standard device-name for the card punch of the 1402 Card Read Punch. "xxx" is the name of a symbolic unit (e.g., MR2).*
PRINTER xxx	is the standard device-name for the 1403 Printer with 132 print positions. "xxx" is the name of a symbolic unit (e.g., MR3).*

Figure 2 illustrates a sample FILE-CONTROL paragraph.

[illegible]

### I-O-CONTROL Paragraph

\*The COBOL programmer may reference symbolic units as either xxx or /xxx/. For details concerning symbolic units see the publication, *System Monitor*.

I-O-CONTROL. [ APPLY *literal-1* PADDING ON *file-name* ]

[ APPLY ... ]

[ APPLY OPEN-WITHOUT-REWIND ON *file-name-2* ]

[ APPLY ... ]

[ RERUN EVERY BEGINNING OF REEL OF

{ ALL FILES  
 *file-name-1* [*file-name-2* ... ] } ].

APPLY *literal-1* PADDING ON *file-name*

Note that spaces or nines should be used for padding characters if the file is to be sorted using the 1410/7010 Generalized Tape Sorting Program.

APPLY OPEN-WITHOUT-REWIND ON *file-name*

NOTE: Both APPLY options can be used for a given file.

***RERUN Option:*** This option allows the programmer to specify rerun points (checkpoints) at every beginning of reel of all files, or of selected files. The tape upon which the rerun records are recorded is the optional Core Image file. (Information concerning checkpoints is contained in the publication, *IBM 1410/7010 Operating System; Basic Input/Output Control System*, Form C28-0322. Information concerning restarting a program from a checkpoint is contained in the publication, *IBM 1410/7010 Operating System; Operator's Guide*, Form C28-0351.)

The Data Division of a COBOL source program defines the nature and characteristics of the data to be processed by the object program. It begins with the header DATA DIVISION. Each of the three sections of the Data Division also begins with a header, and is followed by the word SECTION as shown below:

```
DATA DIVISION.
FILE SECTION.
    File Description Entries
    Record Description Entries
WORKING-STORAGE SECTION.
    Record Description Entries
CONSTANT SECTION.
    Record Description Entries
```

The File Section describes the input/output files with respect to content and organization. It has two types of entries: the File Description entry, which specifies the physical characteristics and organization of a file; and the Record Description entry, which describes the individual items contained in the data records of the file.

The Working-Storage Section describes the areas of core storage where intermediate results and other items are stored temporarily at object-program execution time.

The Constant Section describes fixed items of data which remain unchanged during the running of the object program.

Any section not required in the program being written should be omitted.

### IBM 1410/7010 Files and Records

The programmer should understand how files and records are handled by the IBM 1410/7010 Operating System in order to use the COBOL language effectively in writing the Data Division entries for his source program. Information concerning files and records is therefore given below, prior to discussion of the COBOL language specifications for the Data Division.

#### Recording Modes

Information in a data processing system may be recorded in various forms and modes. The following discussion pertains to the file-recording modes of the IBM 1410 and 7010 Data Processing Systems. For additional details, see the publication, *IBM 1410 Principles of Operation*, Form A22-0526 or *IBM 7010 Principles of Operation*, Form A22-6726.

*ples of Operation*, Form A22-0526 or *IBM 7010 Principles of Operation*, Form A22-6726.

#### EVEN AND ODD PARITY MODES

The IBM 1410 and 7010 can record information on magnetic tape and read information from magnetic tape in either even-parity mode or odd-parity mode.

#### LOAD AND MOVE MODES

Another 1410/7010 file recording mode specifies how word marks and word separator characters are recorded during read and write operations.

*Load Mode:* The handling of word marks and word separator characters in the Load mode depends on the type of operation, as follows:

During *write* operations, each word mark is translated into a word separator character that immediately precedes the character with which the word mark was associated in core storage. Each word separator character in storage is translated into two word separator characters on tape.

During *read* operations, word marks already in the input area are cleared. Each word separator character on tape is translated into a word mark associated with the character it immediately preceded on tape, and pairs of word separator characters on tape are translated into single word separator characters without word marks in core storage.

*Move Mode:* When information is written in the Move mode, word marks have no effect on the data that is recorded on output media. Word marks in storage are undisturbed when information is read in this mode. Each word separator character is read into core storage and written out of core storage as a word separator character.

#### Standard Tape Labels

If STANDARD labels are specified in the File Description entry, certain items within the label are automatically processed by the COBOL compiler. The remaining items may be used by the programmer by using the BEGINNING-LABEL and/or the ENDING-LABEL options of the LABEL RECORDS clause in the File Description entry.

For details concerning the form of the standard tape labels, see the publication, *IBM 1410/7010 Operating System; Basic Input/Output Control System*, Form C28-0322.

## Record Formats for Tape Files

The data record formats that can be handled by the 1410/7010 COBOL compiler for files assigned to tape are:

- 1a. Fixed-length, unblocked records with or without terminal record marks (Figure 3).
- 1b. Variable-length, unblocked records with terminal record marks and without length checking (Figure 4).
2. Fixed-length, blocked records with terminal record marks (Figure 5).
3. Variable-length, unblocked records containing a Record Character-Count field and with or without terminal record marks (Figure 6).

The Record Character-Count field is a four-position field at the beginning of each record. It contains a count of the total number of characters in that record, including itself and the terminal record mark, if present.

4. Variable-length, blocked records with a Block Character-Count field and containing Record Character-Count fields. Terminal record marks are required (Figure 7).

A four-character Block Character-Count (bcc) field at the beginning of each block contains a count of the

total number of characters in the block (including the four-character Block Character-Count field itself).

This count is used to check and correct wrong-length-record conditions. The bcc field must have AB zone bits (12-punch) over the units position.

This field is not a part of a record and therefore is not defined in a Record Description entry.

A Record Character-Count (rcc) field of one to four characters in each record contains a count of the total number of characters in that record, including itself and the terminal record mark. This field must be in the same relative position in each record (the number of characters in each "C1" in Figure 7 is the same), and must be the same length in each record of a given file. The "C2" fields in Figure 7 are all equal in length.

## Record Formats for Unit-Record Files

### CARD READ PUNCH RECORDS

Records of files assigned to the card reader or card punch must be 80 characters in length, unblocked, and may or may not have record marks in the 80th character position. In addition, these files must be in Move mode and even parity.

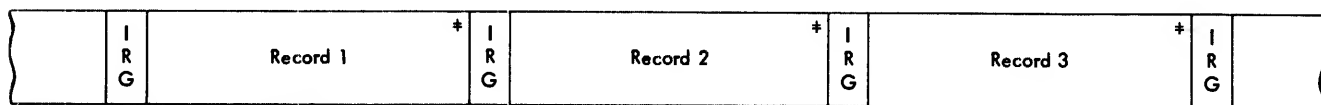


Figure 3. Fixed-Length, Unblocked Records

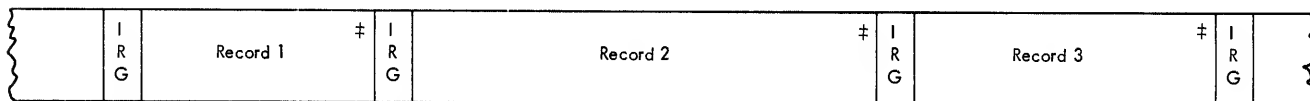


Figure 4. Variable-Length, Unblocked Records Without Length Checking

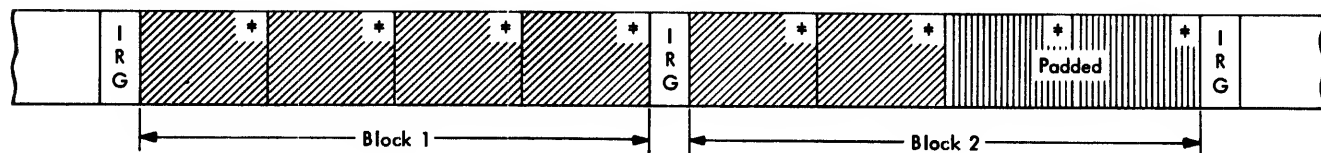


Figure 5. Fixed-Length, Blocked Records

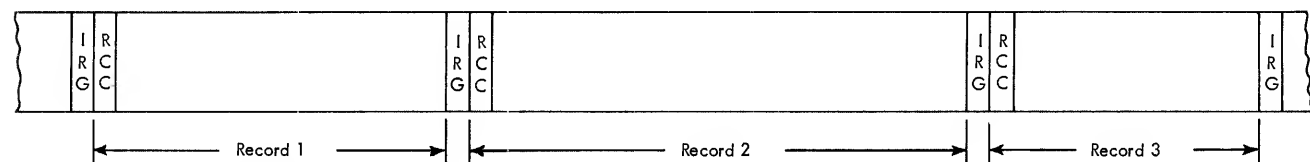


Figure 6. Variable-Length, Unblocked Records

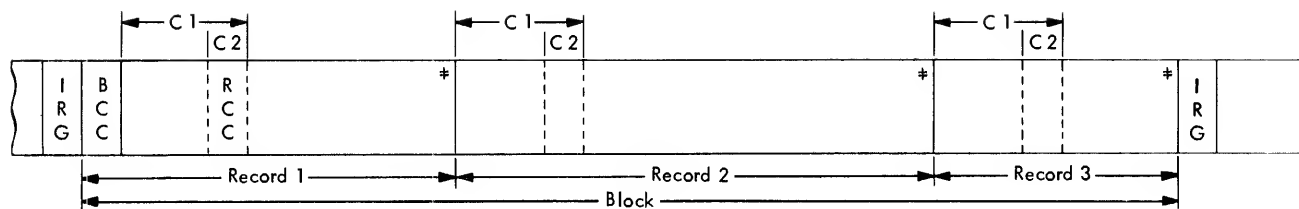


Figure 7. Variable-Length, Blocked Records

#### PRINTER RECORDS

Records of files assigned to the printer must be 132 characters, fixed-length, and unblocked. Files assigned to the printer must be in Move mode and even parity.

## File Section

### File Description Entry

A File Description entry must describe each file to be processed by the object program. It includes specifications for the mode in which the file is recorded, record and block size, label record information, and the names of the data records that make up the file.

The form of the File Description entry is:

```

FD file-name [RECORDING MODE IS
                {MOVE} MODE {EVEN} PARITY
                {LOAD} {ODD}
                [BLOCK CONTAINS integer-1 {RECORDS}
                {CHARACTERS}]
RECORD CONTAINS [integer-2 TO
                  integer-3 CHARACTERS
                  [DEPENDING ON data-name-1]
LABEL RECORD[S] {ARE}
                  {IS}
                  {STANDARD [WITH integer-4 CHARACTERS]
                    [BEGINNING-LABEL]
                    [ENDING-LABEL]
                  }
                  {OMITTED
                  }
                  {NON-STANDARD [WITH integer-4 CHARACTERS]
                    [BEGINNING-LABEL]
                    [ENDING-LABEL]
                  }
                  [VALUE OF FILE-IDENTIFICATION IS literal-1
                  [RETENTION-PERIOD IS integer-5]]
DATA RECORD[S] {ARE}
                {IS} data-name-2 [data-name-3 ...].
  
```

**Level Indicator:** The level indicator FD identifies the beginning of the File Description entry and precedes the file-name assigned by the programmer (Figure 8).

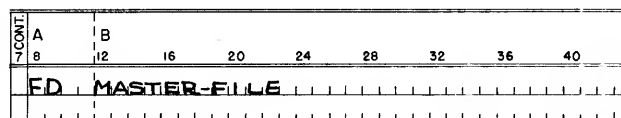


Figure 8. FD File-Name

**RECORDING MODE Option:** This option specifies the mode in which the file is recorded. (See the "Recording Modes" section of this publication.) If the RECORDING MODE option is omitted in the source program, the compiler assumes Move mode and even parity.

**BLOCK CONTAINS Option:** In addition to the details specified in the General Information Manual, the following information pertains to the BLOCK CONTAINS option.

If the file-name in the FD entry contains variable-length records, this entry must take the form:

BLOCK CONTAINS integer-1 CHARACTERS

where integer-1 must be equal to or greater than the number of characters contained in the longest block of the file. This number includes the four-character Block Character-Count (bcc) field (see variable-length, blocked records in the section, "Record Formats for Tape Files").

**RECORD CONTAINS Clause:** This required clause is used to specify the size of a record in terms of the number of characters it contains and to indicate the record form. Integer-2 is used to specify the minimum number of characters in the smallest record of the file, whereas integer-3 indicates the maximum number of characters in the largest record. If all records in the file are exactly the same size, only integer-3 should be specified.

The DEPENDING ON data-name-1 option is required only when specifying variable-length records with Record Character-Count (rcc) fields. Data-name-1 is the name of the rcc field. The contents of this field indicate the number of characters in the record.

The following examples illustrate the use of the `BLOCK CONTAINS` option and the `RECORD CONTAINS` clause to specify each of the five record forms:

For fixed-length, unblocked records:

`RECORD CONTAINS 80 CHARACTERS`

For variable-length, unblocked records without length checking:

`RECORD CONTAINS 100 TO 200 CHARACTERS`

For fixed-length, blocked records:

`BLOCK CONTAINS 5 RECORDS`  
`RECORD CONTAINS 80 CHARACTERS`

For variable-length, unblocked records:

`RECORD CONTAINS 100 TO 200 CHARACTERS DE-`  
`PENDING ON RCC`

For variable-length, blocked records:

`BLOCK CONTAINS 504 CHARACTERS`  
`RECORD CONTAINS 30 TO 50 CHARACTERS DE-`  
`PENDING ON RCC`

***LABEL RECORD Clause:*** This clause is required in every File Description entry. For unit-record files, this clause must specify that label records are `OMITTED`. If `STANDARD` labels are specified for tape files, the file identification, the reel sequence, and the retention period are automatically checked.

If either `STANDARD` or `NON-STANDARD` is specified and the `WITH integer-4 CHARACTERS` option is desired, `integer-4` must be 80 or 120. This is required in order to conform with the 1410 80-character and IBM Standard 120-Character tape labels. (For details concerning these labels see the publication, *Basic Input/Output Control System*.) If this option is not used, the label record size is assumed to be 120 characters.

**NOTE:** Actual size of nonstandard labels need not be exactly 80 or 120 characters, but may not exceed 120.

When a file contains standard tape labels, and no processing beyond that supplied by the compiler is required, `STANDARD` must be specified.

If additional processing of the standard tape label is desired, the programmer must specify `STANDARD` with `BEGINNING-LABEL` and/or `ENDING-LABEL` in conjunction with the `USE` verb. If either or both of these options are used, a Record Description entry that defines the entire label must be provided.

*Example:*

```

.
.
.
LABEL RECORDS ARE STANDARD BEGINNING-LABEL
ENDING-LABEL
.
.
.
```

```
01 BEGINNING-LABEL.
```

```
02...
02...
```

```
01 ENDING-LABEL.
```

```
02...
02...
```

When a file contains nonstandard labels and label processing is not desired, `NON-STANDARD` must be specified. Use of `NON-STANDARD` without additional options will cause the nonstandard labels to be bypassed in the object program.

Special processing of nonstandard labels can be accomplished by defining the label format with the `BEGINNING-LABEL` and `ENDING-LABEL` options in conjunction with the `USE` verb. No automatic testing takes place if `NON-STANDARD` is specified.

***VALUE Option:*** The function of the `VALUE` option in the File Description entry is to specify the contents of data items in the label record of the file. The following two forms of the `VALUE` option are permitted for standard tape labels;

*Form 1.*

`VALUE OF FILE-IDENTIFICATION` IS *literal-1*

This form applies to both input and output files and is required if standard tape labels are used. `Literal-1` must be a ten-character non-numeric literal.

*Form 2.*

`VALUE OF FILE-IDENTIFICATION` IS *literal-1*  
`RETENTION-PERIOD` IS *integer-5*

This form applies to output files only, and must be supplied for each output file if standard tape labels are used. `Integer-5` must be an integer (up to four digits) indicating the number of days beyond the creation date the file is to be preserved. For files that are to be preserved indefinitely, the programmer inserts the digits "99" in the two high-order positions of the creation date (see "Standard Tape Labels").

***DATA RECORD Clause:*** This clause is required in every File Description entry. `Data-name-2`, `data-name-3`, . . . etc., must each be the subject of a Record Description entry that has a level number of 01. The data-name order is not significant to the processor.

The appearance of more than one data-name in this clause means that the file contains a corresponding number of different types of data records. These records may be of different sizes and formats.

Figure 9 illustrates a sample File Description entry.

## Record Description Entry

A Record Description entry specifies to the compiler the characteristics of each item of a data record. Every item given a separate name must be described in a



The general form of this clause is:

[SIGNED]

**VALUE Clause:** The general form of the VALUE clause is:

[VALUE IS *literal*]

In addition to the details specified in the General Information Manual, the following information pertains to the use of this clause:

1. If the VALUE clause specifies a numeric literal with a preceding sign, the operational sign is created only if the programmer specifies the PICTURE symbol "S" or the SIGNED clause.

2. The VALUE clause can only be used to refer to elementary items.

3. The VALUE clause has no meaning for report items, and cannot be used to specify their initial values.

4. Neither a record mark ( $\neq$ ) nor a group mark ( $\equiv$ ) can be used within the VALUE clause (see PICTURE symbols "J" and "K").

**PICTURE Clause:** The general form of the PICTURE clause is:

[PICTURE IS *any allowable combination of*  
*characters and symbols*]

The PICTURE clause can only be used to describe elementary items. It is recommended that, wherever possible, the programmer use this clause instead of the SIZE, POINT, CLASS, and BLANK clauses of a Record Description entry. The PICTURE clause specifies the characteristics of an elementary item in a more compact form, and can therefore be processed more efficiently.

In addition to the rules given in the General Information Manual for forming a picture of a data item, the following information pertains to the use of the PICTURE clause:

1. The only way to define a record mark or group mark is by using a PICTURE symbol. The special PICTURE symbol "J" is used to indicate a one-character field containing a record mark ( $\neq$ ), and the special PICTURE symbol "K" is used to indicate a one-character field containing a group mark ( $\equiv$ ). When used, the PICTURE symbol "J" or "K" must be the only character in the PICTURE.

2. The PICTURE symbol "S" is used to indicate an operational sign (see the SIGNED clause).

3. For report items, the maximum number of characters that can be represented by a PICTURE is 99.

4. The PICTURE symbol "V" to the right or left of PICTURE symbol "P" is redundant and constitutes an error.

5. PICTURE symbol "Z" may appear to the right of a decimal point in a PICTURE only if all numeric character positions are represented by "Z"s. This same rule applies to the replacement character "\*".

**BLANK WHEN ZERO Clause:** The general form of the BLANK WHEN ZERO clause is:

[BLANK WHEN ZERO]

See the General Information Manual for details concerning the use of this clause.

Figure 10 illustrates a sample Record Description entry.

### **Working-Storage and Constant Sections**

The Record Description entries described for the File Section apply also to the Working-Storage and Constant Sections. These sections begin with the header line "WORKING-STORAGE SECTION." or "CONSTANT SECTION." and are followed immediately by the Record Description entries.

In addition to the details specified in the General Information Manual, the following considerations pertain to the Working-Storage and Constant Sections.

If the VALUE clause is not used to define the initial values of Working-Storage items, these values will be unpredictable.

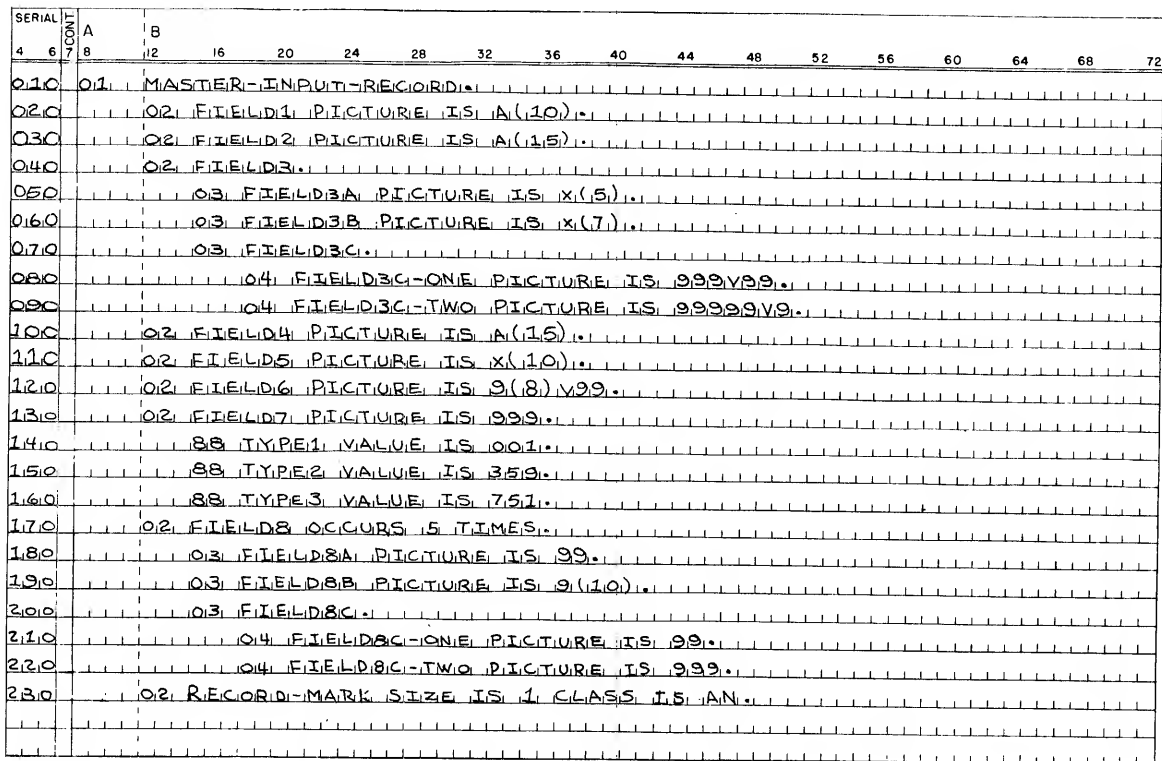
Constant Section elementary items must include a VALUE clause or one of the PICTURE symbols "J" and "K", unless associated with a REDEFINES clause.

### **Added Features of the Data Division**

An optional feature, not specified in the General Information Manual, but contained in the 1410/7010 COBOL language is:

The DEPENDING ON option of the RECORD CONTAINS clause.





Data Division 17

## Procedure Division

Once the computer and the data have been described, the programmer gives the COBOL compiler the instructions that specify the data processing steps the object program is to perform. Information that directs the compiler is also supplied by the programmer in the Procedure Division.

The COBOL verbs are the main elements in the Procedure Division and they are described in detail in the General Information Manual. However, for the convenience of the user, the general format of each COBOL verb is shown in this publication.

Information not included in the General Information Manual, but applicable to verbs which have additional meaning when used in a 1410/7010 COBOL source program, is supplied in later sections of this manual.

## Compiler Directing Declaratives

Declaratives are compiler-directing statements that operate under the control of the “main body” of the Procedure Division or the Input/Output Control System. Declaratives consist of the `USE` verb and any associated procedures. If present, Declaratives

1. must be grouped at the beginning of the Procedure Division, and
2. must be preceded by the key word **DECLARATIVES** and followed by the key words **END DECLARATIVES**.

### USE Verb

The **USE** verb is used to specify procedures for handling input/output errors and label processing (see “**LABEL RECORD**” clause) in addition to the procedures normally provided by the **COBOL** compiler. Each occurrence of the **USE** verb constitutes a complete section of the Procedure Division, and must therefore be preceded by a section-name. The remainder of the section consists of one or more paragraphs specifying the procedures to be used (Figure 11). A “**USE**” section must follow the rules for a section which is “performed” (see the **PERFORM** verb in the General Information Manual for details). For example, if the logic of a procedure requires a conditional exit prior to the final statement, the **EXIT** verb must be used. The section-name of a **USE** section must not be referenced by an **ALTER**, **GO TO**, or **PERFORM** statement since the **USE** entry is appended to this section header. The paragraph-name of the last paragraph in a **USE** section must not be referenced by an **ALTER** or **PERFORM** statement since the **USE** exit is appended to this paragraph.

[illegible]

Figure 11. Declaratives

The following options of the `USE` verb are available.

*Option 1*

*section-name* SECTION. USE AFTER STANDARD ERROR PROCEDURE ON *file-name-1*, [*file-name-2* . . .]. (any COBOL statements including DISPLAY using the console printer but excluding all other input/output statements)

This option allows the programmer to specify additional procedures if standard error-correction procedures do not suffice. Standard error procedures attempt to correct the error in accordance with 1410/7010 rcs standards. If the error persists, the error condition is ignored and processing continues. Thus, the programmer has the following options: ignore the error condition and process the record, or by means of the `USE` verb perform special processing for error records (e.g., set a switch, display a console message, etc.).

### Option 2

<i>section-name</i>	SECTION.	USE	$\left\{ \begin{array}{c} \text{BEFORE} \\ \text{AFTER} \end{array} \right\}$	STANDARD
			$\left[ \left\{ \begin{array}{c} \text{BEGINNING} \\ \text{ENDING} \end{array} \right\} \right]$	$\left[ \left\{ \begin{array}{c} \text{REEL} \\ \text{FILE} \end{array} \right\} \right]$

LABEL PROCEDURE ON *file-name*. (*any COBOL statements including DISPLAY using the console printer but excluding all other input/output statements*)

This option is used to:

1. Perform processing of standard tape labels beyond that supplied by the COBOL compiler.
2. Perform any desired processing of nonstandard labels.

If both **BEGINNING** and **ENDING** are omitted, the designated procedures will be executed for both beginning (header) and ending (trailer) labels.

If both **REEL** and **FILE** are omitted, the designated procedures will be executed upon detection of both end-of-reel and end-of-file conditions.

## Input/Output Verbs

### OPEN and CLOSE

The COBOL compiler provides the facility for opening an input or output file, processing it, closing it, and subsequently reopening it as an input or output file.

The **OPEN** verb is used to initiate the processing of one or more input and/or output files. The format of the **OPEN** verb is:

**OPEN** [ **INPUT** *file-name-1* [ *file-name-2* . . . ] ]  
[ **OUTPUT** *file-name-3* [ *file-name-4* . . . ] ]

The **CLOSE** verb is used to terminate processing of one or more input and/or output reels or files. Provision for optionally locking or not rewinding is also included. The format of the **CLOSE** verb is:

**CLOSE** *file-name-1* [ **REEL** ] [ **WITH** { **LOCK**  
                  **NO REWIND** } ]  
[ *file-name-2* . . . ]

See the General Information Manual for details concerning the **OPEN** and **CLOSE** verbs.

### READ

The function of this verb is to make the next record from an input file available for processing. The general form of the **READ** verb is:

**READ** *file-name* **RECORD** [ **INTO** *area-name* ]  
                  **AT** **END** *any imperative statement*

In addition to the details specified in the General Information Manual, the following considerations pertain to the use of the **READ** verb:

1. An **OPEN** statement for the file must be executed prior to the execution of the first **READ** for that file.
2. When a **READ** is executed, the next record of the file becomes accessible in the input area defined by the

associated Record Description entry in the File Section of the Data Division. The record remains available in the input area until the next **READ** (for that file) is executed. The named file must be defined by an **FD** entry in the Data Division of the program.

3. Every **READ** statement must include an **AT END** clause containing any imperative statements; i.e., any single verb with its operand(s), or a sequence of verbs with their operands terminated by a period and containing no explicit or implied conditional expressions. Once an **AT END** statement has been executed, any attempt to **READ** from the file will constitute an error unless a subsequent **CLOSE** and **OPEN** have been executed.

**NOTE:** When reading a file containing fixed-length, blocked records, the end-of-file condition does not necessarily occur following the last logical record. Therefore, the programmer must test for a record consisting of all padding characters, to ensure detection of the end of the logical file.

4. The **INTO** area-name option converts the **READ** into a **READ** and **MOVE**. The area-name specified must be the name of either a Working-Storage record area or an output record area. When this option is used, the current record becomes available in the input area, as well as in the area specified by area-name. If the format of the **INTO** area differs from that of the input record, the data will be moved in accordance with the rules for the **MOVE** verb without the **CORRESPONDING** option. It will be assumed that the area specified by area-name will be completely filled by information from the input record. If this is not the case, **READ** and **MOVE** should be used rather than **READ INTO**.

5. Each time an end-of-reel condition occurs in a reel other than the last, the **READ** verb causes the following operations to take place:

- a. If labels are present (as specified in the **FD** for that file) the standard end-of-reel label subroutine of the Input/Output Control System is executed.
- b. A tape alternation occurs, if appropriate.
- c. If labels are present, the standard beginning-of-reel label subroutine is executed.
- d. If **RERUN** has been specified for this file, a checkpoint record is written.
- e. The next record in the file is made available for processing.

### WRITE

The purpose of the **WRITE** verb is to release a record for insertion in an output file. The format of a **WRITE** statement is:

**WRITE** *record-name* [ **FROM** *area-name* ]

1. If the user desires to write records which have been described by the `RENAME` option (see the “`FILE CONTROL`” paragraph), the record-name must always be qualified by the file-name.

## DISPLAY

$$\underline{\text{DISPLAY}} \left\{ \begin{matrix} \textit{data-name-1} \\ \textit{literal-1} \end{matrix} \right\} \left[ \left\{ \begin{matrix} \textit{data-name-2} \\ \textit{literal-2} \end{matrix} \right\} \dots \right] \left[ \text{UPON } \textit{mnemonic-name} \right]$$

1. `DISPLAY` literals must be non-numeric.
2. The Operating System's Standard Punch Unit and Standard Print Unit may be equated with mnemonic-names in the `SPECIAL-NAMES` paragraph of the Environment Division. If the `UPON` option is omitted, the console printer will be used as the standard Display Device.

4. If a printer is used, it will be assumed that the carriage tape has a channel-1 punch.

6. A standard set of error procedures is produced by the compiler for use in the execution of the `DISPLAY` verb.

Figure 13 shows a `DISPLAY` statement that will cause the contents of the field `GRAND-TOTAL` to be displayed in 80-character records on the Standard Punch Unit

A	B
8	16      20      24      28      32      36      40
	<b>DISPLAY GRAND-TOTAL.</b>

Figure 12. Standard Display Device

A	B
8	12 16 20 24 28 32 36 40
	DISPLAY GRAND-TOTAL UPON PUNCH.

when the object program is executed, assuming that the mnemonic-name PUNCH has been equated with the SYSTEM-OUTPUT-PUNCH.

The function of the ACCEPT verb is to obtain low-volume data from the Operating System's Standard Input Unit. The Standard Input Unit is the only device from which information can be accepted. The general form of the ACCEPT verb is:

Figure 14 shows an ACCEPT statement that will cause data to be read from the Standard Input Unit and moved into the area defined by the data-name CANCELATIONS. If this area contains more than 80 characters, sufficient card images will be read to fill it.

A	B
8	16      20      24      28      32      36      40
	ACCEPT CANCELLATIONS.

A standard set of error procedures is produced by the compiler for use in the execution of the `ACCEPT` verb.

The **MOVE** verb can be used in either of two formats:

$$\underline{\text{MOVE}} \left\{ \begin{array}{l} \text{data-name-1} \\ \text{literal} \end{array} \right\} \underline{\text{TO}} \text{data-name-2} \quad \left[ \text{data-name-3} \dots \right]$$

MOVE CORRESPONDING *data-name-1* TO  
*data-name-2* [*data-name-3* . . .]

In addition to the details specified in the General Information Manual, the following considerations pertain to the use of Options 1 and 2 of the MOVE verb:

1. The following moves are aligned by decimal point:
  - a. Elementary numeric to elementary numeric
  - b. Elementary numeric to elementary alpha-numeric report
  - c. Elementary non-numeric to elementary numeric (The elementary non-numeric item is assumed to be an integer.)
2. All other moves are left-justified.
3. When figurative constants are used as operands of the MOVE verb, the size of the receiving area determines the number of characters that are moved. For example, if the size of AREA-A in Figure 15 is five positions, its value after execution is five nines (99999).

CONT	A	B																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											</
------	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

Figure 15. Move Verb

4. When using a MOVE CORRESPONDING, only the initial description of data-name-1 and data-name-2, etc., will be considered. That is, where a REDEFINES clause has been used within data-name-1 or data-name-2, the description of data contained within the REDEFINES clause is ignored. (See general suggestions given in the section, "Programming Techniques.")

## EXAMINE

The general form of the EXAMINE verb is:

EXAMINE *data-name*

$$\left\{ \begin{array}{l} \text{TALLYING} \left\{ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{UNTIL FIRST} \end{array} \right\} \text{literal-1} \\ \left[ \text{REPLACING BY literal-2} \right] \\ \text{REPLACING} \left\{ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{UNTIL FIRST} \end{array} \right\} \text{literal-3 BY literal-4} \end{array} \right\}$$

See the General Information Manual for details concerning the EXAMINE verb.

## Arithmetic Verbs

The following rules apply to the arithmetic verbs:

1. All data-names used in arithmetic statements must represent elementary numeric data items that are defined in the Data Division of the program. A data-

name that is defined within the Constant Section cannot appear as the result field of an arithmetic statement.

2. All literals used in arithmetic statements must be numeric.

3. The maximum size of any operand (data-name or literal) is 18 digits. If the format for any operand specifies a size greater than 18 digits, the compiler will produce an error message when it encounters the discrepancy.

4. Intermediate result fields generated for the evaluation of arithmetic expressions (formulas) will always have a PICTURE of S9(10)V9(10). If greater precision is desired, the simple arithmetic verbs (i.e., ADD, SUBTRACT, MULTIPLY, and DIVIDE) must be used.

5. Decimal-point alignment is supplied automatically throughout computations.

6. The format of any data item involved in computations (e.g., addends, subtrahends, multipliers, etc.) cannot contain editing symbols. If this rule is violated, the compiler will indicate the error by an appropriate message. Operational signs and implied decimal points are not considered editing symbols. The data-name in the GIVING option and the result field in the COMPUTE verb format represent data items which must not enter into computations if they contain editing symbols.

7. The only figurative constant permitted in arithmetic statements is ZERO (or ZEROS and ZEROES).

8. For the simple arithmetic verbs the data characteristics of the receiving field control the precision of the operation; therefore, no high-order digit will be lost without creating the SIZE ERROR condition. All specified decimal positions will also be produced.

9. For use with the SIZE ERROR option "any imperative statement" is any single verb with its operand(s) or a sequence of verbs with their operands terminated by a period and containing no explicit or implied conditional expressions.

10. If exponentiation is used in a COMPUTE expression, the exponent must be an integer. Negative exponents are permitted.

## ADD

The general form of the ADD verb is:

$$\text{ADD} \left\{ \begin{array}{l} \text{data-name-1} \\ \text{literal-1} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{data-name-2} \\ \text{literal-2} \end{array} \right\} \dots \right] \\ \left[ \left\{ \begin{array}{l} \text{TO} \\ \text{GIVING} \end{array} \right\} \text{data-name-n} \right] \left[ \text{ROUNDED} \right] \\ \left[ \text{ON SIZE ERROR any imperative statement} \right]$$

An ADD statement must name at least two addends. For additional details concerning the ADD verb, see the General Information Manual.

**CORRESPONDING Option:** The CORRESPONDING option of the ADD verb allows the programmer to specify the addition of corresponding items in one operation in a manner similar to MOVE CORRESPONDING.

The general form of ADD CORRESPONDING is:

ADD CORRESPONDING *data-name-1* TO *data-name-2*

[ROUNDED]  
[ON SIZE ERROR *any imperative statement*]

Numeric elementary items within *data-name-1* are added to numeric elementary items with matching names in *data-name-2*. *data-name-1* and *data-name-2* must be nonelementary items. The rules stated for the simple ADD verb apply to each pair of items in the ADD CORRESPONDING option.

Only the initial description of *data-name-1* and *data-name-2* is considered in the implementation of the ADD CORRESPONDING option. That is, where a REDEFINES clause has been used for *data-name-1* or *data-name-2*, the description of the data contained within the REDEFINES clause is ignored by ADD CORRESPONDING. (See general suggestions given in the section, "Programming Techniques.")

The ROUNDED option and the SIZE ERROR option of the ADD verb may also be used with ADD CORRESPONDING. For a detailed description of these two options, see the General Information Manual.

NOTE: When SIZE ERROR is used in conjunction with CORRESPONDING, the SIZE ERROR test is made only after the completion of all the add operations. If any of the additions produced a SIZE ERROR, the resultant field for that add remains unchanged, and the "any imperative statement" is executed.

To illustrate the use of the ADD CORRESPONDING option, assume that the programmer wishes to add items from a work area named RECEIPTS to corresponding items in an area designated STOCK-ON-HAND. He would write this statement:

ADD CORRESPONDING RECEIPTS TO STOCK-ON-HAND

Figure 16 shows what will result from this statement. Note that noncorresponding items in the STOCK-ON-HAND area are not affected.

## SUBTRACT

The general form of the SUBTRACT verb is:

SUBTRACT {*data-name-1*}  
*literal-1* [ {*data-name-2*}  
*literal-2* ... ]  
FROM {*data-name-n*}  
*literal-n* [ GIVING *data-name-m* ]  
[ROUNDED]  
[ON SIZE ERROR *any imperative statement*]

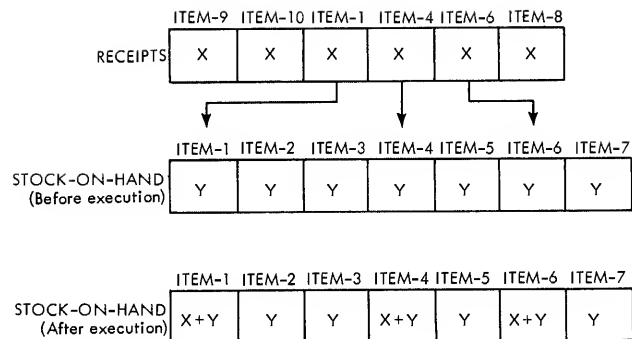


Figure 16. Add Corresponding

A SUBTRACT statement must name at least one subtrahend and one minuend. For further details concerning the SUBTRACT verb, see the General Information Manual.

**CORRESPONDING Option:** The CORRESPONDING option of the SUBTRACT verb functions in the same way as the CORRESPONDING option of the ADD verb.

The general form of SUBTRACT CORRESPONDING is:

SUBTRACT CORRESPONDING *data-name-1*  
FROM *data-name-2* [ROUNDED]  
[ON SIZE ERROR *any imperative statement*]

## MULTIPLY

The general form of the MULTIPLY verb is:

MULTIPLY {*data-name-1*}  
*literal-1* BY {*data-name-2*}  
*literal-2* [GIVING *data-name-3*] [ROUNDED]  
[ON SIZE ERROR *any imperative statement*]

Additional details concerning the MULTIPLY verb are contained in the General Information Manual.

## DIVIDE

The general form of the DIVIDE verb is:

DIVIDE {*data-name-1*}  
*literal-1* INTO {*data-name-2*}  
*literal-2* [GIVING *data-name-3*] [ROUNDED]  
[ON SIZE ERROR *any imperative statement*]

Additional details concerning the DIVIDE verb are contained in the General Information Manual.

## COMPUTE

The general form of the COMPUTE verb is:

COMPUTE *data-name-1* [ ROUNDED ]  
            $\equiv$  *arithmetic expression*  
           [ ON SIZE ERROR *any imperative statement* ]

For details concerning the COMPUTE verb, see the General Information Manual.

## Procedure Branching Verbs

### GO TO

There are two formats in which the GO TO verb can be used:

#### Option 1

GO TO [ *procedure-name* ]

#### Option 2

GO TO *procedure-name-1* *procedure-name-2*  
       [ *procedure-name-3* . . . ] DEPENDING ON *data-name*

For additional information concerning the GO TO verb, see the General Information Manual.

### ALTER

The general form of the ALTER verb is:

ALTER *procedure-name-1* TO PROCEED TO  
           *procedure-name-2* [ *procedure-name-3* TO  
           PROCEED TO *procedure-name-4* . . . ]

A GO TO sentence that is to be altered must be:

1. An unconditional GO TO sentence
2. Written as a separate paragraph consisting solely of the GO TO sentence, preceded by a procedure-name

For additional information concerning the ALTER verb, see the General Information Manual.

### PERFORM

There are five formats in which the PERFORM verb can be used. These are:

#### Option 1

PERFORM *procedure-name-1* [ THRU *procedure-name-2* ]

#### Option 2

PERFORM *procedure-name-1* [ THRU *procedure-name-2* ]  
           { *integer-1* } TIME[S]  
           *data-name-1* }

#### Option 3

PERFORM *procedure-name-1* [ THRU *procedure-name-2* ]  
                                   UNTIL *condition-1*

#### Option 4

PERFORM *procedure-name-1* [ THRU *procedure-name-2* ]  
                                   VARYING *data-name-1* FROM  
                                   { *numeric-literal-1* } BY { *numeric-literal-2* }  
                                   *data-name-2* } *data-name-3* }  
                                   UNTIL *condition-1*

#### Option 5

PERFORM *procedure-name-1* [ THRU *procedure-name-2* ]  
                                   VARYING *subscript-name-1*  
                                   FROM { *integer-1* } BY { *integer-2* }  
                                   *data-name-1* } *data-name-2* }  
                                   UNTIL *condition-1* [ AFTER *subscript-name-2*  
                                   FROM { *integer-3* } BY { *integer-4* } UNTIL  
                                   *condition-2* ] [ AFTER *subscript-name-3* FROM  
                                   { *integer-5* } BY { *integer-6* } UNTIL *condition-3* ]  
                                   *data-name-5* } *data-name-6* }

See the General Information Manual for further information concerning the PERFORM verb.

## Compiler Directing Verbs

### ENTER

The ENTER verb, used in conjunction with the CALL verb, allows the programmer to incorporate into his object program FORTRAN and/or Autocoder compiled subprograms. The incorporation of subprograms is performed at the time the object program is processed by the Linkage Loader. (See the publication, *System Monitor*.) Each ENTER statement must constitute a separate paragraph in the source program.

The form of the ENTER verb is:

ENTER { COMMUNICATION-MODE }  
           { COBOL }

The entry ENTER COMMUNICATION-MODE precedes the calling of the subprogram(s). The CALL verb specifies the subprogram(s) to be included in the object program. The entry ENTER COBOL must terminate the list of subprograms. COMMUNICATION-MODE may be entered any number of times in a program.

### CALL

The general form of the CALL verb is:

CALL *subprogram-1* [ USING { *data-name-1* }  
                                   *literal-1* }  
                                   [ { *data-name-2* } . . . ] ] [ CALL . . . ]  
                                   { *literal-2* } }

Subprogram-1 is the name contained in the TITLE card of the subprogram. The CALL verb causes the COBOL compiler to generate an *imbedded* call for the named subprogram. When the imbedded call is processed by the Linkage Loader, it is converted into a branch to the first character of the called subprogram.

The USING option specifies the required parameters (data-names and/or literals) for the subprogram. These parameters reference data within the COBOL program and are the only means of communication between the main program and the subprogram. At object program run time, these parameters are represented by a sequence of five-character addresses of the appropriate data, with a word mark over the high-order position of each address. This list is followed by a terminal "No Operation" instruction. (The number of parameters is used by the called subprogram to determine the point at which control is to be returned to the main program.) Although any number of parameters may be specified, a maximum of two subscripted data-names may appear in a given USING option.

The CALL verb may be used only after the COMMUNICATION-MODE has been entered. No other verb may appear within the COMMUNICATION-MODE.

## EXIT

The EXIT verb is used when it is necessary to provide an end point for a procedure that is to be executed by means of a PERFORM statement, or for procedures specified in the "USE" section. While EXIT is classified as a compiler-directing verb because it supplies the compiler with necessary information and does not produce any coding in the object program, it can also be thought of as a "dummy" program verb.

EXIT must appear in the source program as a one-word paragraph preceded by a paragraph-name. The form of the EXIT verb is:

EXIT.

Further discussion of the EXIT verb is contained in the General Information Manual.

## NOTE

The form of the NOTE verb is:

NOTE *any comment.*

See the General Information Manual for any additional information concerning this verb.

## Ending Verb

### STOP

The general form of the STOP verb is:

STOP { literal }  
          { RUN }

In addition to the details specified in the General Information Manual, the following information pertains to the use of the STOP verb:

1. The statement:

STOP *literal*

will cause the program to print the literal on the console printer and enter the Wait-Loop routine of the Resident Monitor. (For details, see the publication, *System Monitor*.)

2. The statement:

STOP RUN

indicates the end of the program and generates the message "STOP RUN" on the console printer, followed by a return to the System Monitor.

## Conditional Expressions

In addition to the details contained in the General Information Manual, the following rule applies to conditional expressions:

Within a relational expression the subject, relational operator, and object must all be at the same logical parenthetical level. Therefore, a left parenthesis preceding an object indicates that arithmetic follows.

*Example:*

VALID	INVALID
IF A = (B + C)	IF A = (B OR C)
IF A = (- B)	IF A = ( (B + C) OR D)

As described in the General Information Manual, implied subjects and implied relational operators are permissible in conditional expressions. No other abbreviated usage is permitted.

*Example:*

VALID	INVALID
IF A = B OR C	IF A = B, C OR D

In a conditional expression the logical operator NOT is only permitted at one given parenthetical level.

*Example:*

VALID	INVALID
NOT (A OR B)	NOT (NOT A OR B)

## Added Features of the Procedure Division

The following features, not contained in the General Information Manual, are included in the 1410/7010 COBOL language:

1. The USE Declarative
2. The CORRESPONDING option of the ADD verb
3. The CORRESPONDING option of the SUBTRACT verb



### Programming Techniques

As mentioned in the General Information Manual, COBOL provides a convenient method of writing business-oriented programs. However, certain techniques can be used which may produce more efficient machine-language coding or increase compiling speed.

The following suggestions are included to aid the user in obtaining the most efficient machine-language coding from the 1410/7010 COBOL compiler:

1. For files which contain multiple records it may be more economical to define only one form and then transfer the record to an appropriate work area.
2. In an ADD or SUBTRACT operation where there are several operands (data-name or literal), the operands should be the same size. If this is not possible, the largest operand should appear first.
3. It is important to use signed rather than unsigned numeric fields wherever possible.
4. For elementary numeric items, the scaling variations should be minimized (use of PICTURE symbol "P").
5. Subscripting and REDEFINES clause usage may be less efficient than other approaches.
6. Whenever possible, simple statements referencing elementary items should be used, rather than complex statements or statements which reference group items.

The following suggestions are included to aid the user in increasing compilation speed:

1. Unnecessary paragraph-names should be avoided.
2. Certain EXEQ card options (see the section "1410/7010 COBOL Compiler Requirements") cause the compiler to produce additional output. When not essential, these options should not be elected.
3. It is recommended that, wherever possible, the programmer use the PICTURE clause instead of the SIZE, POINT, CLASS, and BLANK clauses of a Record Description entry. The PICTURE clause specifies the characteristics of an elementary item in a more compact form, and can therefore be processed more efficiently.

Some general suggestions, which may be beneficial to the programmer, are given below:

1. When desired precision of results of arithmetic expressions exceeds that represented by PICTURE S9(10)V9(10), it is suggested that the appropriate arithmetic verbs be used (i.e., ADD, SUBTRACT, MULTIPLY, and DIVIDE), rather than the COMPUTE verb.

2. The normal contents of the MONITOR-SWITCH, in the Resident Monitor's Communication Region, is a blank. Therefore, it is recommended that the user either: (a) not assign a blank value to a meaningful condition of this switch; or (b) let the blank value indicate that the switch has not been set.

3. Since Load mode deals with word marks, the user should remember the following, when reading or writing tape in Load mode:

- a. Reading a file in Load mode, processing it, and writing the file in Load mode should present no problems, if the files have been described properly.
- b. Writing in Load mode from information that has been read in other than Load mode, may cause unwanted word marks to appear in the output area. (See "WRITE FROM.")
- c. When a REDEFINES clause is associated with a Load mode input file, the input file must be created using a 1410/7010 COBOL object program. (Because of the REDEFINES technique, the redefined portion of the Load mode record does not carry word marks on tape.)

4. The READ verb with the INTO area-name option should not be used if the record is smaller in size than the area specified by area-name. In this case a simple READ followed by a MOVE should be used.

5. Care should be taken when using the CORRESPONDING option if data-name-1 or data-name-2 is associated with a REDEFINES clause at its own level, or if data-name-1 is one of multiple records defined in an input file. In either case, the description of data contained within subsequent entries where a REDEFINES clause has been used is *not* ignored and will affect the results of the statement.

### Compatibility Considerations

Certain COBOL verbs and their associated language specifications cannot be defined in compatible terms between the 1410/7010 Systems and other systems. It is suggested that the user avoid the following when writing COBOL programs that are to be compiled on more than one system:

1. ACCEPT
2. UPON option of the DISPLAY verb
3. ENTER
4. USE

For reasons of compatibility, the use of the **REDEFINES** clause should be limited to one level of redefinition, with the exception that, if the **REDEFINES** is specified at the 01-level, one additional level of redefinition within the level 01 may be used.

Use of the COBOL Character Set (H2) for literals is suggested, when compatibility with other systems is a consideration.

The following clauses described in the General Information Manual are not implemented by the 1410/7010 COBOL compiler for reasons of compatibility:

1. The **JUSTIFIED** clause. Standard justification according to **CLASS** definition will always take place. If nonstandard data manipulation is required, the programmer can use other language specifications for this purpose (e.g., the **REDEFINES** clause).

2. The **Editing** clause. Editing functions can only be specified by use of the **PICTURE** clause.

### Qualification of Names

Every name used in a COBOL source program must be unique within the source program, either because no other name has the identical spelling, or because the name exists within a hierarchy of names (so that the name can be made unique by mentioning one or more of the higher levels of the hierarchy). The higher levels are called qualifiers when used in this way, and the process is called qualification.

In addition to the information contained in the General Information Manual covering the qualification of names, the programmer should note the following:

1. Any name which requires qualification, but is not qualified, will refer to the first occurrence of that name in the program.

2. A name plus all its qualifiers cannot exceed a total of 300 characters. If it does, an error message is produced.

### Literals

In addition to the rules for forming literals specified in the General Information Manual, the following rules apply to the 1410/7010 COBOL compiler:

*For Forming Numeric Literals:* A numeric literal must consist of at least one, and not more than 18 digits. It may also include a sign, preceding the first digit, and/or one decimal point.

*For Forming Non-Numeric Literals:* Any character in the character set, except the quotation mark, the record mark, and the group mark, can be used in a non-numeric (alphanumeric) literal. Blanks are treated as characters and may be included freely.

### Character Sets

The IBM Character Set H2 must be used for COBOL source programs. This character set consists of the numerals 0 through 9, the 26 letters of the alphabet, and 12 special characters. The IBM 1410/7010 Character Set may be used only for alphanumeric literals, with the following exceptions: (1) the IBM 1410/7010 character "␣" (substitute blank) cannot be used with even-parity tape records; (2) the IBM 1410/7010 character "␣" (word separator character) cannot be loaded into the IBM 1410 or 7010 with a word mark.

The COBOL (Set H2) special characters are shown below with their equivalents in the IBM 1410/7010 Character Set:

CARD CODE	COBOL (SET H2)	1410/7010 (SET A2)	MEANING
blank			space
11	—	—	{ minus sign
12	+	&	{ hyphen
0-1	/	/	plus sign
11-4-8	*	*	division sign
12-4-8	)	□	{ multiplication sign
0-4-8	(	%	{ check protection symbol
0-3-8	,	,	right parenthesis
11-3-8	\$	\$	left parenthesis
12-3-8	.	.	comma
3-8	=	#	{ period
4-8	'	@	{ decimal point
			equal sign
			quotation mark

### Figurative Constants

In addition to the details specified in the General Information Manual, the following information pertains to the figurative constants. All figurative constants are treated as belonging only to the **ALPHANUMERIC** class.

LOW-VALUE	The value of this figurative constant is the space, or blank, the lowest in the collating sequence.
LOW-VALUES	
HIGH-VALUE	This figurative constant is defined as the character 9, the highest in the collating sequence.
HIGH-VALUES	
ZERO	This figurative constant represents the value 0. It is the only figurative constant that can be treated as belonging to the <b>NUMERIC</b> class or the <b>ALPHANUMERIC</b> class.
ZEROS	
ZEROES	
SPACE	This figurative constant represents a blank, or space. It is the only figurative constant that can be treated as belonging to the <b>ALPHABETIC</b> class or the <b>ALPHANUMERIC</b> class.
SPACES	
QUOTE	This figurative constant represents the character '. Note that the use of the word <b>QUOTE</b> to represent the character ' is not equivalent to the use of symbol ' to bound a literal.
ALL "literal"	This figurative constant generates a sequence of characters specified by the single-character non-numeric literal.

## TALLY

The word **TALLY** is the name of a data item whose **PICTURE** is **S99999**. It is used primarily to hold information produced by the **EXAMINE** verb; however, it may be referenced by the programmer in any statement where a signed numeric field is valid.

## MONITOR-DATE

In addition to the figurative constants, the IBM 1410/7010 COBOL compiler provides the programmer with the special data-name constant **MONITOR-DATE**. This data-name constant is the name of a five-character data item (system symbol **/DAT/**) within the Communication Region of the Resident Monitor. **MONITOR-DATE** contains the current date established by the System Monitor, and may be used in label-checking routines. The form of the date is **yyddd**, where: **yy** is the year (00-99) and **ddd** is the day of the year (001-366). **MONITOR-DATE** can be used in the same way as any item described in the Constant Section.

## Class Conditions

The General Information Manual specifies that the **CLASS** of a data item may be **NUMERIC**, **ALPHABETIC** or **ALPHANUMERIC**. It further specifies that the class condition is used to test an **ALPHANUMERIC** item at object

time to determine whether it is wholly numeric or wholly alphabetic in content.

The source statement beginning:

IF **FIELD-A** IS **NUMERIC**...

results in a character-by-character check of the value of **FIELD-A** at object time. If an operational sign is present in the units position, the associated character will be interpreted as being numeric. Thus, **-9** is interpreted as "minus 9," not as the letter "R."

The source statement beginning:

IF **FIELD-B** IS **ALPHABETIC**...

results in a character-by-character check of the value of **FIELD-B** at object time. If each character in **FIELD-B** is alphabetic, the item is considered alphabetic.

*Examples:* The following table shows how the class of an item is interpreted by the compiler depending on which of the class tests is specified:

CHARACTER	NUMERIC	ALPHABETIC
0-9	YES	NO
SPACE	NO	YES
A-R	YES (if units position)	YES
S-Z	NO	YES
? !	YES (if units position)	NO
Other		
Special		
Characters	NO	NO

## 1410/7010 COBOL Compiler Requirements

It is assumed in the following material that the user is thoroughly familiar with the contents of the publication, *System Monitor*.

### Requirements for Compilation

In order to process a COBOL source program under the IBM 1410/7010 Operating System, certain control cards are required to direct the operation of the Resident and Transitional Monitors and the Linkage Loader. The required Monitor control cards are:

```
MON$$ JOB
MON$$ MODE
MON$$ EXEQ
MON$$ ASGN
```

The required Linkage Loader control cards are:

```
PHASE
CALLN
CALL
```

These control cards are described in detail in the publication, *System Monitor*. However, certain COBOL options, which are available to the user, are discussed below.

### EXEQ Card Operand Options

The user can control the output of the COBOL compiler by placing operands immediately after the comma which follows the third System Monitor option on the EXEQ card. These operands can appear in any order and must be separated by commas, with no intervening blanks. Any of the following operands may be used:

1. **LIST**—This operand produces a listing of source program names and corresponding object program relocatable storage assignments. A check for duplicate procedure-names is made if this option is elected. A warning message appears if duplicate names are present.
2. **DIAGNOSTIC**—This operand suppresses the creation of an object program. (DIAGNOSTIC cannot be requested on the same EXEQ card with TRACE or NOPCH.)
3. **TRACE**—This operand causes the generation of a self-tracing object program. When each paragraph or section of the main body of the Procedure Division is executed at object time, the paragraph or section-name is printed on the Standard Print Unit.
4. **NOPCH**—This option should be used only when a Go file is being created. The function of NOPCH is to

suppress output on the Standard Punch Unit, thereby providing an object program on the Go file only.

In the event of an error in the use of any of these options on the EXEQ card, the compiler will ignore all options, and produce only the normal output (an object program on the Go file and/or the Standard Punch Unit).

Figure 17 shows an EXEQ card for COBOL compilation with the TRACE and LIST options.

Line	Label	Operation		OPERAND					
		15	16	20	21	25	30	35	40
01	MON\$\$	EXEQ	COBOL	TRACE	LIST				
02									

Figure 17. EXEQ Card for COBOL Compilation

### Requirements for Execution

The object program produced by the COBOL compiler consists of several subprograms. In accordance with the requirements of the Linkage Loader, each subprogram is headed by a TITLE card.

#### The Subprogram TITLE Card

The COBOL compiler generates all necessary TITLE card information based on the source program.

The format of the TITLE card is:

```
Column   6      16    21          31  36      73
          yyddd  TITLEPROGRAMnnnnxxxxxxxxzzzzz  nnnsssss
```

where

*yyddd* is the current date taken from the Resident Monitor's Communication Region.

*PROGRAM* is the first seven characters of the IDENT field of the PROGRAM-ID card in the source program.

*nnn* is the subprogram number, assigned serially by the compiler. This number is placed in columns 28-30 and columns 73-75.

*xxxxx* is the lowest relocatable storage address occupied by the subprogram.

*zzzzz* is the character count of common storage used by the subprogram.

*sssss* is the sequence-number field of the cards (or card images) in the subprogram. The sequence number of each TITLE card will always be 00001.

#### IDENT Field of the PROGRAM-ID Card

In order to comply with 1410/7010 Operating System requirements, the following restrictions pertain to completing the IDENT field:

1. It must always begin with an alphabetic character.

2. It cannot begin with the characters "IB".
3. It cannot contain the slash (/) or any blank characters.

If these requirements are not met, the compiler will replace the erroneous character with "A". For example, if the IDENT field contains IBPSD/b/, the TITLE cards will contain IAPSDAANN.

### Multiple Subprogram COBOL Output

The following list shows the subprogram serial number and function in the normal output of a compilation:

SERIAL NO.	SUBPROGRAM FUNCTION
001	Storage allocation and value declarations for Identification, Environment, and Data Divisions
002	Storage allocation and value declarations for Procedure Division literals
003	Object code for Procedure Division
004	Overlay addresses

### Control Card Requirements

The sequence of the appropriate Monitor and Linkage Loader control cards needed to compile and execute

the program with the IDENT "PAYROLL1" using the TRACE option is shown in Figure 18.

The subprogram IBCOBOL is a required part of every object program. It must be requested with a CALLN card immediately after the PHASE card.

COBOL programs that have been compiled can be added to the System Library file. (For details, see the publication, *System Monitor*.) The CALL requirements for executing these programs from the System Library file are the same as those for the Go file.

Immediately following the CALLN card for IBCOBOL (Figure 18) is a CALL card for the first of the just-compiled subprograms. The name used in this card consists of the first seven characters of the IDENT field (PAYROLL) and the serial number, 001. The other three subprograms (PAYROLL002, PAYROLL003, and PAYROLL004) are processed by the Linkage Loader in response to *imbedded* calls that the compiler generates for each set of subprograms.

The Linkage Loader places the relocated program on the Job file, from which it is loaded by the Resident

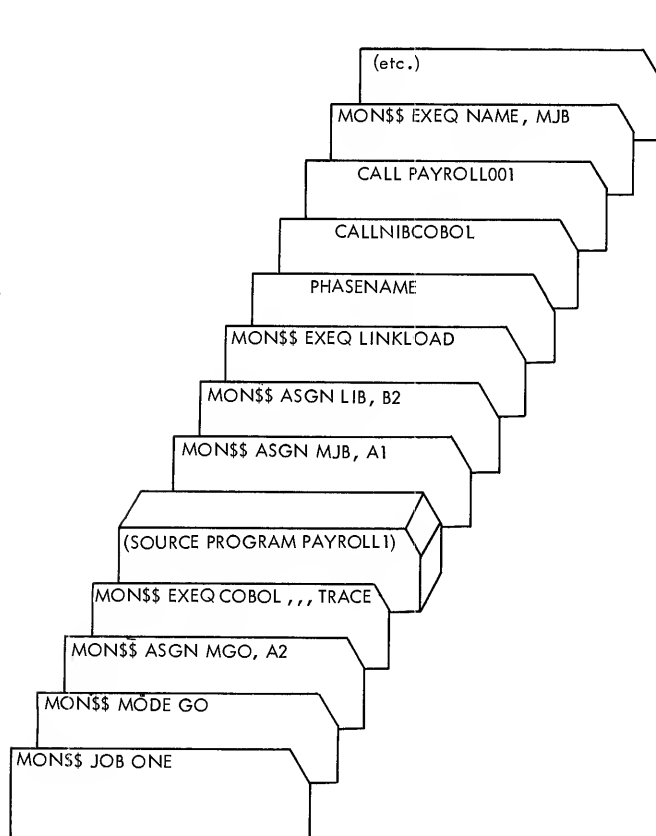


Figure 18. Sample Control Cards for a Compile-and-Go Operation

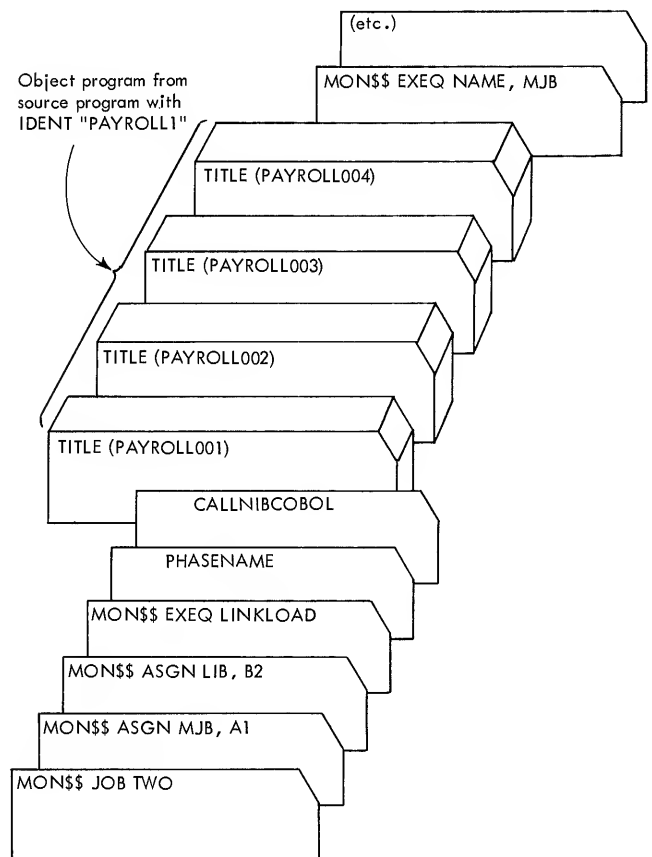


Figure 19. Sample Control Cards for Execution

Monitor (EXEQ NAME, MJB). Note that the name used in the EXEQ card for the program must be the same as that used in the PHASE card given the Linkage Loader, but need not be the same as that used in the IDENT field given the compiler.

The sequence of the appropriate Monitor and Linkage Loader control cards necessary to “execute” the program with the IDENT “PAYROLL”, compiled at a *prior* time is shown in Figure 19.

In Figure 19, the object program has been taken from the Standard Punch Unit and is submitted to the Linkage Loader from the Standard Input Unit (rather than from the Go file, as in Figure 18). The basic difference between the two examples (Figures 18 and 19) is that in Figure 19 a CALL card is not used for the subprogram PAYROLL001, because the TITLE card of a subprogram placed in the Standard Input Unit serves the call function.

## Appendix A: COBOL Words

The words listed below constitute the complete IBM COBOL vocabulary. Words preceded by an asterisk (\*) are not implemented by the 1410/7010 COBOL compiler but should be avoided when assigning names to data, etc., to avoid unnecessary difficulty in converting 1410/7010 COBOL programs to other IBM systems.

Programmers are cautioned that the words recognized by the 1410/7010 COBOL compiler can be used in a COBOL source program only as specified in this publication, or in the General Information Manual.

ACCEPT	CHARACTERS	DIVISION	INSTALLATION
ADD	*CHECKPOINT-UNIT	*ELECTRONIC-SWITCH	INTO
*ADDRESS	CLASS	*ELIMINATION	I-O-CONTROL
*ADDRESSES	CLOSE	ELSE	*IOCS
AFTER	COBOL	END	*IOHSK
ALL	*COLLATE-MACHINE-	ENDING	I-O-SWITCH
ALPHABETIC	SEQUENCE	*ENDING-FILE	IS
ALPHANUMERIC	COMMUNICATION-	ENDING-LABEL	LABEL
ALTER	MODE	ENDING-REEL	LEADING
ALTERNATE	COMPUTATIONAL	ENTER	LEFT
AN	*COMPUTATIONAL-1	ENVIRONMENT	LESS
AND	*COMPUTATIONAL-2	EOF-SIU	*LIBRARY
APPLY	COMPUTE	EQUAL	LOAD
ARE	CONFIGURATION	ERROR	LOCATION
AREA	CONSOLE-PRINTER	EVEN	LOCK
AREAS	*CONSOLE-SWITCH	EVERY	*LONG-LENGTH-RECORD
ASSIGN	CONSTANT	EXAMINE	*LOW
AT	*CONTAIN	EXIT	LOW-VALUE
AUTHOR	CONTAINS	FD	LOW-VALUES
AUTHORS	*CONTROLS	FILE	*MEMORY
	*COPY	FILES	MODE
*BCD	CORRESPONDING	FILE-CONTROL	MONITOR-DATE
BEFORE	*CREATION-DATE	FILE-IDENTIFICATION	MONITOR-SWITCH
BEGINNING	*CREATION-DAY	FILLER	MOVE
BEGINNING-LABEL	*CREATION-YEAR	FIRST	*MULTIPLE
*BEGINNING-REEL		FOR	MULTIPLY
*BINARY	DATA	FROM	
BLANK	DATE-COMPILED		NEGATIVE
BLOCK	DATE-WRITTEN	GIVING	NEXT
*BLOCKS	DECLARATIVES	GO	NO
BY	*DENSITY	GREATER	*NO-LENGTH-CHECK
	DEPENDING		*NONE
CALL	DIGIT	*HEADER-LABEL	NON-STANDARD
CARD-PUNCH	DIGITS	*HIGH	*NO-OVERLAP
CARD-READER	DISPLAY	HIGH-VALUE	*NO-PRINT-STORAGE
CHARACTER	DIVIDE	HIGH-VALUES	*NO-RELEASE
		*HYPERTAPE-UNIT	NOT
		*HYPERTAPE-UNITS	*NO-TAPE-MARK
			NOTE
		IBM-1410	NUMERIC
		IBM-7010	
		IDENTIFICATION	OBJECT-COMPUTER
		IF	*OBJECT-PROGRAM
		IN	OCCURS
		INPUT	ODD
		INPUT-OUTPUT	OF

OFF	*SHORT-ALPHA-WORD
OMITTED	*SHORT-LENGTH-RECORD
ON	SIGNED
OPEN	SIZE
OPEN-WITHOUT-REWIND	SOURCE-COMPUTER
*OPTIONAL-USAGE	SPACE
OR	SPACES
OTHERWISE	SPECIAL-NAMES
OUTPUT	STANDARD
	STATUS
PADDING	STOP
PARITY	SUBTRACT
PERFORM	*SUPERVISOR
PICTURE	SYNCHRONIZED
PLACE	*SYSTEM-INPUT-UNIT
PLACES	SYSTEM-OUTPUT-
POINT	PRINTER
POSITIVE	SYSTEM-OUTPUT-PUNCH
*PREASSEMBLED	
PRINTER	TALLY
*PRIORITY	TALLYING
PROCEDURE	TAPE-UNIT
PROCEED	*TAPE-UNITS
PROGRAM-ID	THAN
*PROGRAM-START	THEN
	THROUGH
QUOTE	THRU
QUOTES	TIME
	TIMES
READ	TO
RECORD	*TRAILER-LABEL
RECORDING	*TYPEWRITER
*RECORD-MARK	
RECORDS	*UNIT-RECORD-I-O-
REDEFINES	RECORD
REEL	UNTIL
*REELS	UPON
*REEL-SEQUENCE-	USAGE
NUMBER	USE
*REFERENCE	USING
REMARKS	
RENAMING	VALUE
REPLACING	VARYING
RERUN	
RESERVE	WHEN
RETENTION-PERIOD	WITH
REWIND	*WITH-LABELS
RIGHT	*WITHOUT-LABELS
ROUNDED	*WORDS
RUN	WORKING-STORAGE
	WRITE
SECTION	
SECURITY	ZERO
SELECT	ZEROES
SENTENCE	ZEROS

## Appendix B: Organization of Source Program

Some items which may appear in a source program are required, while others are optional. Whether an item is required or optional may be determined by reading the discussion of each individual COBOL word in this publication. The order of appearance of the divisions is mandatory and all divisions must be present. Certain sections within the divisions must also appear as specified, while others have no rigid rules. The items which may appear in a source program are the following:

IDENTIFICATION DIVISION.  
PROGRAM-ID. *program-name*.  
AUTHOR. *author-name*.  
INSTALLATION. ...  
DATE-WRITTEN. ...  
DATE-COMPILED. ...  
SECURITY. ...  
REMARKS. ...

ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER. ...  
OBJECT-COMPUTER. ...  
SPECIAL-NAMES. ...  
INPUT-OUTPUT SECTION.  
FILE-CONTROL. SELECT ...  
I-O-CONTROL. APPLY ...

DATA DIVISION.  
FILE SECTION.  
FD *file-name-1* ...  
    01 *data-name-1* ...  
        02 *data-name* ...  
            03 *data-name* ...  
            88 *condition-name* ...

        .  
        .  
        .  
    02 *data-name* ...  
        .  
        .  
    01 *data-name* ...  
        .  
        .  
        .

FD *file-name-2* ...

        .  
        .  
        .  
FD *file-name-n* ...

WORKING-STORAGE SECTION.  
77 *data-name* ...  
    88 *condition-name* ...

        .  
        .  
        .  
77 *data-name* ...

        .  
        .  
        .  
01 *data-name* ...  
    02 *data-name* ...

        .  
        .  
        .



```

01 data-name ...
  02 data-name ...
    03 data-name ...
    88 condition-name ...
      .
      .
      .
02 data-name ...
  .
  .
  .
01 data-name ...
  .
  .
  .
CONSTANT SECTION.
77 data-name ...
  .
  .
  .
77 data-name ...
01 data-name ...
  02 data-name ...
    .
    .
    .
01 data-name ...
  02 data-name ...
    03 data-name ...
      .
      .
      .
    02 data-name ...
      .
      .
      .
01 data-name ...
  .
  .
  .
PROCEDURE DIVISION.
DECLARATIVES.
section-name SECTION. USE ...
paragraph-name. ...
END DECLARATIVES.
paragraph-name. ...
  .
  .
  .
section-name-1 SECTION.
paragraph-name-1. ...
  .
  .
  .
paragraph-name-2. ...
  .
  .
  .
paragraph-name-n. ...
  .
  .
  .

```

```

section-name-2 SECTION.
  .
  .
  .
section-name-n SECTION.
paragraph-name-1. ...
  .
  .
  .
paragraph-name-2. ...
  .
  .
  .
paragraph-name-n. ...
  .
  .
  .

```

## Appendix C: Object Time Error Analysis and Messages

The following object time conditions will cause immediate job termination. The messages will appear on the Standard Print Unit and control will be transferred to the Resident Monitor's Unusual End of Program. (For details see the *System Monitor* publication.)

### INVALID EXPONENTIATION

An attempt to raise zero to the zero power has been detected.

### SUBSCRIPTING ERROR

A subscript that is zero, negative, or out of range (of the array), has been detected.

### INVALID COMPLEX PERFORM

Object program has failed to follow COBOL rules for PERFORM. To locate this error it is suggested that the program be recompiled and executed using the TRACE option.

### ZERO DIVISOR

An attempt to divide by zero has been detected.

### UNALTERED STATEMENT nnnnnn

A GO TO statement which required ALTERing was executed prior to being ALTERed.

If the program being run was compiled in the TRACE mode, the name of the paragraph in which this error occurred appears as the last paragraph-name on the Standard Print Unit.

If the program being run was *not* compiled in the TRACE mode, nnnnnn is the relocated address of the unaltered GO TO statement.

## Appendix D: Diagnostic Messages

This appendix includes all the diagnostic messages produced by the 1410/7010 COBOL compiler, and their meanings. The messages are listed by division, with a "general" section for messages which can occur in more than one division.

Normally, when a diagnostic message appears on the source program listing, an incomplete object program will be produced. The compiler will continue to examine the entire source program for further errors but terminates object program output. However, some messages are merely warnings to the programmer, and do not necessarily affect the compilation of the source program. A "W" preceding a message in this appendix indicates a warning-type message. Also indicated (for warning-type messages) are any assumptions the compiler may make about the intent of the statement in question.

### Source Program Listing

The following information is included to assist the programmer to better understand the source program listing:

1. An "S" appearing to the left of a statement, or group of statements, indicates that the programmer-supplied sequence numbers are out of sequence. (This is a warning and does not affect compilation.)
2. The four-digit number appearing on the extreme left of the source program Procedure Division listing is a card reference number assigned by the compiler.
3. Source statements and diagnostic messages for the Identification, Environment, and Data Divisions appear interspersed in the source program listing.
4. Diagnostic messages for the Procedure Division appear in one section of the source program listing.
5. A message will appear indicating the storage allocated to the main program. This allocation does not include any optional subprograms called by the user or by the compiler.
6. One of the following messages appears on the Standard Print Unit if compilation of the source program has been prevented:

#### a. SHORT LENGTH WORK FILE (I/O operation)

Appears if one of the work files (MW1, MW2, or MW3) assigned for the COBOL compiler is not of sufficient length. Compilation is terminated and control passes to the Resident Monitor's Special-End-of-Program routine. (See the publication, *System Monitor*, for details.)

#### b. \*\*\*\*\* OBJECT PROGRAM INCOMPLETE \*\*\*\*\*

Appears if there are errors in the source program which cannot be corrected by the COBOL compiler. It is the terminating message of a COBOL compilation in

which a source error prevented the generation of a complete object program, unless DIAGNOSTIC has been specified on the EXEQ card. (See 7, below.) If this is a compile-and-go operation, the Go file is cancelled.

#### c. UNCORRECTABLE I/O ERROR IN (phase name)

Appears if there is some uncorrectable input or output error. Compilation is terminated and control passes to the Resident Monitor's Special-End-of-Program routine. (See the publication, *System Monitor*, for details.)

#### d. \*\*\*\*\* SOURCE PROGRAM INCOMPLETE \*\*\*\*\*

Appears if the four COBOL Divisions are not found in the source program. Compilation is terminated and control passes to the Resident Monitor's Special-End-of-Program routine. (See the publication, *System Monitor*, for details.) If this is a compile-and-go operation, the Go file is cancelled.

In conjunction with messages a, b, and c, the following messages appear on the console printer:

10980 SHORT LENGTH WORK FILE (I/O operation)

10990 OUTPUT INCOMPLETE—SOURCE ERROR

10999 OUTPUT INCOMPLETE—I/O ERROR

(phase name)

7. The following message appears on the Standard Print Unit, and is the terminating message for a COBOL "DIAGNOSTIC" run.

\*\*\*\*\* END OF DIAGNOSTIC RUN \*\*\*\*\*

### General

The messages appearing below can occur in more than one division.

- |   |                                                                                                                                                                                   |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| W | <b>DUPLICATE CLAUSE KEYWORD</b><br>A clause keyword has appeared more than once in an entry.                                                                                      |
| W | <b>INVALID CONTINUATION CARD</b><br>Blank continuation card. It is ignored.                                                                                                       |
| W | <b>INVALID LITERAL SYNTAX</b><br>Record mark or group mark found in a VALUE clause or a non-numeric literal format error.                                                         |
| W | <b>KEYWORD DIVISION MISSING</b><br>Presence of the word DIVISION is assumed.                                                                                                      |
| W | <b>REFERENCE FORMAT ERROR</b><br>One of the format rules has been broken but is ignored by the compiler. (See the General Information Manual.)                                    |
| W | <b>SYNTAX CHECK DISCONTINUED WITH "X"</b><br>The syntactical form of the input statements does not conform to COBOL syntax. "X" is the word initiating the erroneous source data. |
| W | <b>SYNTAX CHECK RESUMED WITH "X"</b><br>A valid syntactical form is recognized after the occurrence of the above message. "X" is the word with which checking is resumed.         |

## Identification Division

- W INVALID IDENTIFICATION ASSIGN IDENTIFICATION "X"  
Columns 73-80 invalid. Value of "X" is assigned by the compiler.
- W SEARCH FOR IDENTIFICATION  
The keyword IDENTIFICATION has not been discovered in its proper position.
- W SEARCH FOR PROGRAM-ID  
The keyword PROGRAM-ID has not been discovered in its proper position.

## Environment Division

- W ASSIGN CLAUSE MISSING  
Self-explanatory.
- W CONFIGURATION SECTION MISSING  
Self-explanatory.
- W CONFIGURATION SECTION OUT-OF-ORDER  
Self-explanatory.
- W DUAL OBJECT-COMPUTER SPECIFICATION  
IBM 1410 and IBM 7010 specified.
- W DUAL SOURCE-COMPUTER SPECIFICATION  
IBM 1410 and IBM 7010 specified.
- W FILE-CONTROL PARAGRAPH MISSING  
Self-explanatory.
- W INPUT-OUTPUT SECTION MISSING  
Self-explanatory.
- W INVALID APPLY CLAUSE SYNTAX  
The clause is ignored.
- W INVALID APPLY LITERAL  
More than one padding character, or invalid padding character.
- W INVALID ASSIGN CLAUSE SYNTAX  
The clause is ignored.
- W INVALID DEVICE-NAMES CLAUSE SYNTAX  
The clause is ignored.
- W INVALID LABEL RECORD SIZE  
Beginning-Label or Ending-Label record is greater than 120 characters.
- W INVALID RENAMING CLAUSE SYNTAX  
The clause is ignored.
- W INVALID RERUN CLAUSE SYNTAX  
The clause is ignored.
- W INVALID RESERVE CLAUSE SYNTAX  
The compiler assumes there are NO ALTERNATE AREAS.
- W INVALID SELECT SYNTAX  
SELECT not followed by ASSIGN, RESERVE, or RENAMING.
- W INVALID SPECIAL-NAMES PARAGRAPH SYNTAX  
The paragraph is ignored.
- W INVALID SWITCH-NAMES CLAUSE SYNTAX  
The clause is ignored.

- W KEYWORD SECTION MISSING  
Self-explanatory.
- W MISSING PERIOD  
Self-explanatory.
- W NO I-O-CONTROL PARAGRAPH  
Self-explanatory.
- W NO SECTION HEADING  
Self-explanatory.
- W OBJECT-COMPUTER PARAGRAPH MISSING  
Self-explanatory.
- W PARAGRAPH INVALID IN THIS SECTION  
Processing will take place as if SECTION were correct.
- W PARAGRAPH OUT-OF-ORDER  
Self-explanatory.
- W SOURCE-COMPUTER PARAGRAPH MISSING  
Self-explanatory.
- W UNDEFINED APPLY FILE-NAME  
File used in APPLY clause not defined.
- W UNDEFINED RENAMING FILE-NAME  
File used in RENAMING clause not defined.
- W UNDEFINED RERUN FILE-NAME  
File used in RERUN clause not defined.

## Data Division

- W 77-LEVEL OUT-OF-ORDER  
Self-explanatory.
- W 88-LEVEL INVALID AT GROUP LEVEL  
Self-explanatory.
- W 88-LEVEL INVALID IN THIS SECTION  
An 88-level appears in the CONSTANT SECTION. This entry is ignored.
- W CLAUSE MISSING IN THIS FD  
LABEL RECORDS clause is missing and is assumed to be omitted; or DATA RECORDS clause is missing and is ignored.
- W ENTRY EXCEEDS MAXIMUM CLASS SIZE  
Indicates a numeric item with a size greater than 18 digits. Class is assumed to be alphanumeric.
- W FD ENTRY RECORD MISSING  
An FD entry has no associated Record Description items. File is ignored.
- W FD OUT-OF-ORDER  
FD has been detected in other than File Section. Compiler will handle this condition.
- W FILE SECTION OUT-OF-ORDER  
Will be processed as if in proper order.
- W INCOMPATIBLE BLOCK RECORD CLAUSE  
Combination of BLOCK CONTAINS and RECORD CONTAINS clause does not agree with one of the five allowable formats.
- W INCOMPATIBLE BLOCK RECORD SIZE  
The record size is too large. Record size will be used.

W	<b>INCOMPATIBLE CLASS PICTURE CLAUSE</b> Classes as specified by the CLASS and PICTURE clauses in a given item do not agree. CLASS clause is ignored.	W	<b>INVALID LABEL RECORD CLAUSE SYNTAX</b> Compiler assumes OMITTED.
W	<b>INCOMPATIBLE LITERAL</b> CLASS or PICTURE does not agree with VALUE literal. The compiler ignores this condition and allocates storage for the literal.	W	<b>INVALID LEVEL-NUMBER</b> The level-number of the first item following an FD is not 01. This item is assumed to be a 01-level.
W	<b>INCOMPATIBLE PICTURE POINT CLAUSE</b> The assumed decimal point in the POINT clause does not agree with the PICTURE. POINT clause is ignored.	W	<b>INVALID LEVEL-NUMBER SYNTAX</b> Invalid level-number sequence. Will be treated as if valid; therefore, hierarchical relationships may be affected.
W	<b>INCOMPATIBLE POINT CLASS CLAUSE</b> The POINT clause is not associated with a numeric item. POINT clause is ignored.	W	<b>INVALID LITERAL</b> File Identification value is improper. If more than 10 characters the value is truncated.
W	<b>INCOMPATIBLE RECORD SIZE</b> The record size as derived from the Record Description does not agree with the size as stated in the RECORD CONTAINS clause. The computed record size will be used.	W	<b>INVALID LITERAL IS THIS CONTINUATION CARD</b> Continuation indicator, but first non-blank character, not the quote sign. Continuation ignored-literal is terminated by end of first card.
W	<b>INCOMPATIBLE REDEFINES ENTRY</b> The size associated with the redefinition is not equal to the size of the original area. The size of the original is used if the redefined area is greater.	W	<b>INVALID OCCURS CLAUSE</b> OCCURS clause generates a fourth or higher dimension array. The clause is ignored.
W	<b>INCOMPATIBLE SIGNED PICTURE CLASS CLAUSE</b> The existence of a sign, specified by the SIGNED clause, does not agree with the PICTURE, which is non-numeric. The SIGNED clause is ignored.	W	<b>INVALID OCCURS CLAUSE SYNTAX</b> The clause is ignored.
W	<b>INCOMPATIBLE SIZE CLAUSE AT GROUP LEVEL</b> SIZE as specified at group level does not agree with the size as calculated from the contained elementary items. Group SIZE is made to conform.	W	<b>INVALID PERIOD</b> Self-explanatory.
W	<b>INCOMPATIBLE SIZE PICTURE CLAUSE</b> Size as specified in a SIZE clause does not agree with the size given by the PICTURE clause. SIZE clause is ignored.	W	<b>INVALID PICTURE SYNTAX</b> The clause is ignored.
W	<b>INCOMPATIBLE WITH HIGHER LEVEL CLASS</b> Class specified for this item does not agree with class specified for the group. Group CLASS is ignored.	W	<b>INVALID POINT CLAUSE SYNTAX</b> The clause is ignored.
W	<b>INVALID 88-LEVEL</b> 88-level occurs without a preceding condition variable (valid level-number). It is ignored.	W	<b>INVALID PUNCTUATION OR SPECIAL CHARACTER</b> This is ignored.
W	<b>INVALID BLOCK CLAUSE SYNTAX</b> The compiler will infer block size from record size.	W	<b>INVALID RECORD SYNTAX</b> Syntactical error in RECORD CONTAINS clause. The clause is ignored.
W	<b>INVALID CLASS SYNTAX</b> CLASS clause is ignored.	W	<b>INVALID RECORDING MODE CLAUSE SYNTAX</b> Compiler assumes Move mode and even parity.
W	<b>INVALID DATA RECORD CLAUSE SYNTAX</b> The clause is ignored.	W	<b>INVALID REDEFINES CLAUSE SYNTAX</b> The redefined data name is undefined or the entries redefining an area do not immediately follow the original definition of the area, or the redefined data-name level-number does not agree with the current-name level-number.
W	<b>INVALID DEPENDING ON ENTRY</b> The DEPENDING ON data name within a given RECORD CONTAINS clause either does not occur in a subsequent file record, or does not have consistent specifications in a multi-record file.	W	<b>INVALID SIZE CLAUSE SYNTAX</b> The clause is ignored.
W	<b>INVALID EDITING CLAUSE SYNTAX</b> Invalid BLANK WHEN ZERO clause.	W	<b>INVALID SYNCHRONIZED CLAUSE SYNTAX</b> The clause is ignored.
		W	<b>INVALID U/R SPECIFICATION</b> Recording mode specified is invalid for unit record. Move mode and even parity is assumed.
		W	<b>INVALID USAGE CLAUSE SYNTAX</b> The clause is ignored.
		W	<b>INVALID VALUE CLAUSE</b> VALUE and REDEFINES clauses are in same item. VALUE and OCCURS clauses are in same item. VALUE in item subordinate to grouped REDE-

- FINES item. VALUE in item subordinate to grouped OCCURS item. VALUE within a File Section Record Description; or VALUE with report item. The VALUE is ignored.
- W KEYWORD SECTION MISSING  
The word SECTION does not appear. The compiler assumes that it is present.
- W LITERAL EXCEEDS MAXIMUM CHARACTER SIZE 120  
Literal will be truncated.
- W LITERAL TRUNCATION  
VALUE exceeds SIZE. This message will also appear whenever a VALUE is given to a field whose PICTURE includes PICTURE symbol "P" on the left.
- W NO CONTINUATION CARD INVALID LITERAL  
No terminal quote sign on current card, or no continuation indicator on next one. Literal assumed terminated at end of first card.
- W NO ENTRY CLASS  
No CLASS or PICTURE for an elementary item. Low order character(s) of the literal will not fit in the field as specified.
- W NO LITERAL WITH 88-LEVEL  
88 is assigned a value of blanks.
- W NO SIZE IN THIS ENTRY  
Self-explanatory.
- NUMBER OF ENTRIES WITHIN GROUP EXCEEDS TABLE SIZE—BREAK UP GROUP USING REDEFINES OPTION  
Self-explanatory.
- W OCCURS CLAUSE INVALID IN THIS ENTRY  
OCCURS clause associated with a 01 or 77-level item. The clause is ignored.
- W PICTURE CLAUSE INVALID AT GROUP LEVEL  
PICTURE clause is describing a group item rather than an elementary item. The clause is ignored.
- W POINT CLAUSE INVALID AT GROUP LEVEL  
POINT clause is used to describe group rather than elementary item. The clause is ignored.
- W PUNCTUATION INVALID IN THIS ENTRY  
One of the punctuation rules has been broken. (See the General Information Manual.) Punctuation is ignored.
- W RECORD CLAUSE MISSING  
RECORD CONTAINS clause is missing.
- RECORD OUT-OF-ORDER  
A Record Description entry within the File Section has no associated FD. Item is processed as WORKING-STORAGE.
- W REDEFINES CLAUSE OUT-OF-ORDER  
REDEFINES clause is not the first clause in an item. The clause is accepted.
- W REDUNDANT 88-LEVEL CLAUSE  
A clause other than VALUE is associated with an 88-level item. This is ignored.
- W SIGNED CLAUSE INVALID AT GROUP LEVEL  
SIGNED clause is used to describe group item rather than elementary item. This is ignored.
- W UNDEFINED DATA-RECORD  
01 Record not defined in DATA RECORD clause.
- W UNDEFINED ENTRY  
Undefined name in REDEFINES clause or RECORD CONTAINS DEPENDING ON clause.
- W UNDEFINED FILE  
FD entry has no associated SELECT clause, or invalid SELECT clause.
- W VALUE CLAUSE INVALID AT GROUP LEVEL  
The clause is ignored.
- W WORD EXCEEDS MAXIMUM CHARACTER SIZE 30  
The word is truncated.
- W WORKING STORAGE SECTION OUT-OF-ORDER  
This is processed as if in proper order.
- ### Procedure Division
- (name) IS AN INVALID QUALIFIER IN (name)  
Invalid qualifier is identified by the paragraph in which it is used.
- (name) IS AN UNDEFINED NAME IN (name)  
Undefined procedure-name is identified by the paragraph in which it is used.
- (name) NOT A CONDITION-NAME  
Self-explanatory.
- (name) OVERSIZE PARAGRAPH  
Paragraph should be broken down into more than one paragraph.
- W CONDITIONAL CLASS CONTRADICTION  
Data items of unlike class are being compared, or a non-numeric data item is being tested for a sign, or a sign test on an unsigned numeric data item.
- W CORRESPONDING OPERATOR INVALID REPLACED WITH "X" OPERATOR  
In MOVE, ADD, or SUBTRACT CORRESPONDING, TO or FROM was missing. "X" will be either "TO" or "FROM."
- CORRESPONDING STATEMENT FORMAT ERROR  
Self-explanatory.
- W CORRESPONDING VERB IGNORED  
CORRESPONDING used with other than MOVE, ADD, or SUBTRACT.
- EXAMINE OPERAND ERROR  
Attempt to EXAMINE a constant or a literal.
- GO TO STATEMENT MISSING DEPENDING  
Self-explanatory.
- INCORRECT CONDITIONAL EXPRESSION  
Self-explanatory.
- W INCORRECT CONTINUATION  
Continuation card error. Text starts prior to column 12 of continuation card. Unnecessary continuation indicator detected. This condition is ignored.
- W INCORRECT END DECLARATIVES  
The compiler will correct this error.
- INCORRECT LITERAL  
Invalid record mark or group mark.
- INCORRECT LITERAL CONTINUATION  
Non-numeric literal continuation error.

W	INCORRECT PUNCTUATION Incorrect punctuation will be ignored.		INVALID STATEMENT Missing ENTER COBOL.
W	INVALID ALTER STATEMENT Something other than a Paragraph/Section-name follows ALTER; TO PROCEED TO is not specified properly; Paragraph/Section-name does not follow TO PROCEED TO; invalid format for compound ALTER statements or more than one level of qualification has been given for Paragraph/Section-name.		INVALID USE STATEMENT Self-explanatory.
	INVALID CALL STATEMENT Self-explanatory.		INVALID WORD AFTER OPEN VERB Self-explanatory.
	INVALID CHARACTER 1410/7010 special character meaningless to COBOL will be ignored.		IS UNDEFINED Undefined name. The name will appear on the preceding line.
	INVALID COMPUTE OPERAND Self-explanatory.	W	LITERAL EXCEEDS 120 CHARACTERS Literal will be truncated.
	INVALID COMPUTE OPERATOR Self-explanatory.		MISSING AT END IN READ STATEMENT Self-explanatory.
	INVALID CONDITIONAL OPERAND Data-name is used incorrectly.		MISSING BY AFTER VARYING IN PERFORM STATEMENT Self-explanatory.
	INVALID CONDITIONAL OPERATOR Self-explanatory.		MISSING DISPLAY OPERAND ONE Self-explanatory.
	INVALID CORRESPONDING CORRESPONDING option is used incorrectly.		MISSING ERROR AFTER SIZE Self-explanatory.
	INVALID DATA-NAME IN ACCEPT STATEMENT Self-explanatory.		MISSING FIRST MOVE OPERAND Self-explanatory.
	INVALID DECLARATIVES Section-name does not follow DECLARATIVES. The compiler will skip to the next procedure-name or END DECLARATIVES.		MISSING FROM AFTER VARYING IN PERFORM STATEMENT Self-explanatory.
	INVALID DISPLAY DEVICE Self-explanatory.		MISSING IF TO MATCH THIS NEXT SENTENCE CLAUSE Self-explanatory.
	INVALID ENTER STATEMENT Self-explanatory.		MISSING LEFT PARENTHESIS IN CONDITIONAL Self-explanatory.
	INVALID EXAMINE STATEMENT Self-explanatory.		MISSING LITERAL IN EXAMINE STATEMENT Self-explanatory.
	INVALID EXIT Keyword EXIT appeared in other than a one-word paragraph.		MISSING LITERAL TWO AFTER EXAMINE Self-explanatory.
	INVALID OPERAND Invalid operands, invalid qualification, or more than three levels of subscripting.	W	MISSING OPERAND ONE IN THIS STATEMENT Self-explanatory.
	INVALID OPERAND AFTER GIVING CLAUSE Multiple receiving fields invalid.		MISSING PERIOD BEFORE P/S NAME Statement not properly terminated before new paragraph/section-name.
	INVALID OPERAND USAGE IN CORRESPONDING Multiple receiving field specified in ADD or SUBTRACT CORRESPONDING, or invalid data-name, such as literal or elementary item used, or a level 77 used.		MISSING PERIOD OR SECTION AFTER PROCEDURE-NAME Self-explanatory.
	INVALID PARENTHESIS Self-explanatory.		MISSING PROCEDURE IN USE STATEMENT Self-explanatory.
	INVALID PERFORM STATEMENT Self-explanatory.		MISSING PROCEDURE-NAME AFTER GO TO Self-explanatory.
			MISSING PROCEDURE-NAME IN PERFORM STATEMENT Self-explanatory.
			MISSING RECEIVING OPERAND Self-explanatory.

MISSING REPLACING OR BY IN EXAMINE STATEMENT Self-explanatory.		MISSING VALID OPERAND AFTER VARYING IN PERFORM Self-explanatory.
MISSING REWIND AFTER NO Self-explanatory.		MISSING VALID READ AREA-NAME Self-explanatory.
MISSING RIGHT PARENTHESIS IN CONDITIONAL Self-explanatory.		MISSING VALID WRITE AREA-NAME Self-explanatory.
MISSING RUN OR LITERAL AFTER STOP Self-explanatory.		MISSING VALID WRITE RECORD Self-explanatory.
MISSING SECOND MOVE OPERAND Self-explanatory.	W	MOVE CLASS CONTRADICTION Self-explanatory.
MISSING SENTENCE AFTER NEXT Self-explanatory.		MOVE OPERAND ERROR The receiving field designated is a literal, constant, etc.
MISSING STATEMENT 1 TO MATCH THIS OTHERWISE OR ELSE The word ELSE or OTHERWISE is used without an associated IF statement.		MOVE SUBSCRIPT FROM OPERAND More than two subscripted data-names have appeared in the USING option of the CALL verb.
MISSING TALLYING OR REPLACING IN EXAMINE STATEMENT Self-explanatory.		NO MATCH FOR CORRESPONDING No match found for MOVE (one item elementary) or Arithmetic (both items elementary numeric). Improper qualifications exists for matching data items. Matching data items are in secondary redefined area, or qualified by same.
MISSING TIMES IN PERFORM STATEMENT Self-explanatory.		PARAGRAPH/SECTION INCOMPLETE IN (name) This message occurs if a source error has prevented processing of part of a statement or paragraph; or a statement implies the existence of a clause or statement that is not present.
MISSING TO AFTER GO Self-explanatory.		W POSSIBLE TRUNCATION Sending or FROM data-name larger than receiving data-name, or storing of arithmetic results where digits might be lost. (This message may occur where the ROUNDING option is used, and should be ignored.)
MISSING TO AFTER MOVE Self-explanatory.		P/S NAME FORMAT ERROR Procedure-name not followed by SECTION or period.
MISSING UNTIL AFTER VARYING IN PERFORM STATEMENT Self-explanatory.		QUALIFIED NAME EXCEEDS STORAGE ALLOCATION Total number of characters has exceeded 300.
MISSING VALID EXPONENTIATE OPERAND Self-explanatory.		SUBSCRIPT ERROR Subscripting used with a data-name not associated with an OCCURS clause, or the number of subscripts used does not agree with the associated data description.
MISSING VALID FILE-NAME AFTER CLOSE VERB Self-explanatory.		USE VERB MISSING In DECLARATIVES, first word after section-name SECTION must be USE. Compiler will skip to the next procedure-name or END DECLARATIVES.
MISSING VALID FILE-NAME AFTER OPEN INPUT Self-explanatory.		W WARNING (name) IS A MULTIPLE DEFINED NAME This message only appears if the LIST option is used and a duplicate name occurs.
MISSING VALID FILE-NAME AFTER OPEN OUTPUT Self-explanatory.		W WORD EXCEEDS 30 CHARACTERS Word is truncated.
MISSING VALID FILE-NAME IN READ STATEMENT Self-explanatory.		
MISSING VALID GO TO DEPENDING OPERAND Data-name is missing or is not an integer.		
MISSING VALID OPERAND AFTER BY OR INTO Self-explanatory.		
MISSING VALID OPERAND AFTER EXAMINE Self-explanatory.		
MISSING VALID OPERAND AFTER GIVING Self-explanatory.		

## Appendix E: Sample Problem

IBM		COBOL PROGRAM SHEET		Form No. 278-1464 Printed in U.S.A.																																				
PAGE 1 3	PROGRAM SAMPLE PROBLEM 1410/7010 COBOL	SYSTEM 1410	SHEET 1 OF 7																																					
001	PROGRAMMER	DATE	IDENT. 72 PAYROLL 2																																					
SERIAL	<table border="1"> <tr> <td>4</td><td>6</td><td>8</td><td>10</td><td>12</td><td>14</td><td>16</td><td>18</td><td>20</td><td>22</td><td>24</td><td>26</td><td>28</td><td>30</td><td>32</td><td>34</td><td>36</td><td>38</td><td>40</td><td>42</td><td>44</td><td>46</td><td>48</td><td>50</td><td>52</td><td>54</td><td>56</td><td>58</td><td>60</td><td>62</td><td>64</td><td>66</td><td>68</td><td>70</td><td>72</td> </tr> </table>					4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72
4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72						
000	IDENTIFICATION DIVISION.																																							
010	PROGRAM-ID. SAMPLE 1410/7010 COBOL PROGRAM.																																							
020	AUTHOR. IBM PROGRAMMING SYSTEMS.																																							
030	REMARKS. DESIGNED TO ILLUSTRATE THE GENERAL FORM OF A COBOL-																																							
040	PROGRAM.																																							
050																																								
060																																								
070	ENVIRONMENT DIVISION.																																							
080																																								
090	CONFIGURATION SECTION.																																							
100	SOURCE-COMPUTER. IBM-1410.																																							
110	OBJECT-COMPUTER. IBM-1410.																																							
120	SPECIAL-NAMES. SYSTEM-OUTPUT-PRINTER IS PRINTER1.																																							
130	I-O-SWITCH EOF-SIM ON STATUS IS LAST-CARD.																																							
140	INPUT-OUTPUT SECTION.																																							
150	FILE-CONTROL.																																							
160	SELECT OUTPUT-PAY-FILE ASSIGN TO TAPE-UNIT MM1.																																							
170	SELECT LIST-FILE																																							
180	ASSIGN TO TAPE-UNIT MM1.																																							
190	I-O-CONTROL.																																							
200	APPLY '9' PADDING ON OUTPUT-PAY-FILE.																																							
210																																								
220																																								

IBM		COBOL PROGRAM SHEET		Form No. 278-1464 Printed in U.S.A.																																				
PAGE 1 3	PROGRAM SAMPLE PROBLEM 1410/7010 COBOL	SYSTEM 1410	SHEET 2 OF 7																																					
002	PROGRAMMER	DATE	IDENT. 72 PAYROLL 2																																					
SERIAL	<table border="1"> <tr> <td>4</td><td>6</td><td>8</td><td>10</td><td>12</td><td>14</td><td>16</td><td>18</td><td>20</td><td>22</td><td>24</td><td>26</td><td>28</td><td>30</td><td>32</td><td>34</td><td>36</td><td>38</td><td>40</td><td>42</td><td>44</td><td>46</td><td>48</td><td>50</td><td>52</td><td>54</td><td>56</td><td>58</td><td>60</td><td>62</td><td>64</td><td>66</td><td>68</td><td>70</td><td>72</td> </tr> </table>					4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72
4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72						
000	DATA DIVISION.																																							
010																																								
020	FILE SECTION.																																							
030	FD OUTPUT-PAY-FILE																																							
040	BLOCK CONTAINS 10 RECORDS																																							
050	RECORD CONTAINS 89 CHARACTERS																																							
060	LABEL RECORDS ARE OMITTED																																							
070	DATA RECORD IS EMPLOYEE-RECORD.																																							
080																																								
090	01 EMPLOYEE-RECORD.																																							
100	02 DISPLAY-RECORD.																																							
110	03 FILLER SIZE IS 5.																																							
120	03 EMPLOYEE-CODE.																																							
130	04 MAN-NUMBER PICTURE IS 9(6).																																							
140	04 FILLER SIZE IS 2.																																							
150	04 MAN-NAME PICTURE IS A(20).																																							
160	04 FILLER SIZE IS 2.																																							
170	04 DEPT-CODE PICTURE IS 99.																																							
180	04 FILLER SIZE IS 5.																																							
190	04 HOURS-WORKED PICTURE IS 99.																																							
200	03 FILLER SIZE IS 5.																																							
210	03 CODE-CONT.																																							
220	04 LABOR-GRADE PICTURE IS 99.																																							
230	04 SHIFT PICTURE IS 9.																																							



PAGE 1		PROGRAM		SYSTEM		SHEET	
PROGRAMMER		DATE		IOENT.		73	
SAMPLE PROBLEM		141017010 COBOL		1410		3 OF 7	
003		003		003		003	
000		03 FILLER SIZE IS 5.					
010		03 GROSS-PAY PICTURE IS \$ZZZ.99.					
020		03 FILLER SIZE IS 5.					
030		03 PREMIUM-PAY PICTURE IS \$ZZZ.99.					
040		03 FILLER SIZE IS 5.					
050		03 TOTAL-PAY PICTURE IS \$ZZZ.99.					
060		02 RM PICTURE IS X.					
070							
080		FD LIST-FILE					
090		BLOCK CONTAINS 10 RECORDS					
100		RECORD CONTAINS 89 CHARACTERS					
110		LABEL RECORDS ARE OMITTED					
120		DATA RECORD IS EMPLOYEE-RECORD1.					
130							
140		01 EMPLOYEE-RECORD1.					
150		02 DISPLAY-RECORD1 CLASS IS AM SIZE IS 88.					
160		02 RM1 PICTURE IS X.					
170							
180		WORKING-STORAGE SECTION.					
190		77 GROSS PICTURE IS 999V99.					
200		77 SHIFT-PREM PICTURE IS 999V99.					
210		77 TOTAL PICTURE IS 999V99.					

PAGE 3		PROGRAM		SYSTEM		SHEET	
PROGRAMMER		DATE		IOENT.		73	
SAMPLE PROBLEM		141017010 COBOL		1410		4 OF 7	
004		004		004		004	
000		01 INPUT-RECORD.					
010		02 EMPLOYEE-CODE.					
020		03 MAN-NUMBER PICTURE IS 9(6).					
030		03 MAN-NAME PICTURE IS A(20).					
040		03 DEPT-CODE PICTURE IS 99.					
050		03 FILLER SIZE IS 2.					
060		03 HOURS-WORKED PICTURE IS 99.					
070		02 CODEIN.					
080		03 LABOR-GRADE PICTURE IS 99.					
090		03 SHIFT PICTURE IS 9.					
100		88 FST VALUE IS 1.					
110		88 SECOND VALUE IS 2.					
120		88 THIRD VALUE IS 3.					
130		88 NO-PREMIUM VALUE IS 4.					
140							
150		01 HOURLY-RATE-TABLE.					
160		02 TRAINEE PICTURE IS 9V99 VALUE IS 1.50.					
170		02 BEGINNER PICTURE IS 9V99 VALUE IS 1.65.					
180		02 JUNIOR PICTURE IS 9V99 VALUE IS 2.00.					
190		02 OPERATOR PICTURE IS 9V99 VALUE IS 2.25.					
200		02 SENIOR PICTURE IS 9V99 VALUE IS 2.65.					
210		02 ASSOCIATE PICTURE IS 9V99 VALUE IS 2.85.					
220		02 STAFF PICTURE IS 9V99 VALUE IS 3.00.					

IBM

## COBOL PROGRAM SHEET

Form No. 228-1464  
Printed in U.S.A.

PAGE	PROGRAM	SYSTEM	SHEET
1	SAMPLE PROBLEM 1410/7010 COBOL	1410	5 OF 7
005	PROGRAMMER	DATE	IDENT 73 PAYROLL 2
SERIAL	A	B	
4 6	8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72		
000	01	RATE REDEFINES HOURLY-RATE-TABLE	
010	02	HRLY-RATE PICTURE IS 9V99 OCCURS 7 TIMES	
020			
030		CONSTANT SECTION	
040	77	RMK PICTURE IS J	
050	77	FIRST-SHIFT PICTURE IS V99 VALUE IS 10	
060	77	SECOND-SHIFT PICTURE IS V99 VALUE IS 13	
070	77	THIRD-SHIFT PICTURE IS V99 VALUE IS 15	
080	77	ZERO-PREM PICTURE IS 999V99 VALUE IS 000.00	
090			
100	01	HEADING-RECORD	
110	02	FILLER SIZE IS 5	
120	02	MNO PICTURE IS A(6) VALUE IS 'MAN-NO'	
130	02	FILLER SIZE IS 5	
140	02	NN PICTURE IS A(4) VALUE IS 'NAME'	
150	02	FILLER SIZE IS 14	
160	02	DPT PICTURE IS A(4) VALUE IS 'DEPT'	
170	02	FILLER SIZE IS 3	
180	02	HRS PICTURE IS A(3) VALUE IS 'HRS'	
190	02	FILLER SIZE IS 4	
200	02	CD PICTURE IS A(4) VALUE IS 'CODE'	
210	02	FILLER SIZE IS 5	
220	02	GROSS PICTURE IS A(5) VALUE IS 'GROSS'	

IBM

## COBOL PROGRAM SHEET

Form No. 228-1464  
Printed in U.S.A.

PAGE	PROGRAM	SYSTEM	SHEET
1	SAMPLE PROBLEM 1410/7010 COBOL	1410	6 OF 7
006	PROGRAMMER	DATE	IDENT 73 PAYROLL 2
SERIAL	A	B	
4 6	8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72		
000	02	FILLER SIZE IS 4	
010	02	PREM PICTURE IS A(7) VALUE IS 'PREMIUM'	
020	02	FILLER SIZE IS 8	
030	02	TOTL PICTURE IS A(5) VALUE IS 'TOTAL'	
040			
050		PROCEDURE DIVISION	
060			
070		DECLARATIVES	
080		UNREADABLE SECTION USE AFTER STANDARD ERROR PROCEDURE ON	
090		LIST FILE DISPLAY DISPLAY-RECORD1	
100		END DECLARATIVES	
110			
120		INTRODUCTION NOTE THAT THIS PROGRAM HAS BEEN DESIGNED	
130		TO DEMONSTRATE TYPICAL COBOL FORMAT NO ATTEMPT HAS BEEN	
140		MADE TO CREATE A PROGRAM FOR ACTUAL CUSTOMER APPLICATION	
150			
160		START OPEN OUTPUT OUTPUT-PAY-FILE	
170		NEXT-EMPLOYEE ACCEPT INPUT-RECORD IF LAST-CARD GO TO REND	
180		MOVE CORRESPONDING EMPLOYEE-CODE IN INPUT-RECORD TO EMPLOYEE-	
190		CODE IN DISPLAY-RECORD MOVE CODEIN TO CODEOUT MULTIPLY	
200		HOURS-WORKED IN INPUT-RECORD BY HRLY-RATE (LABOR-GRADE IN	
210		CODEIN) GIVING GROSS MOVE GROSS TO GROSS-PAY	
220			

IBM

## COBOL PROGRAM SHEET

Form No. X28-1444  
Printed in U.S.A.

PAGE 1	PROGRAM SAMPLE PROBLEM	SYSTEM 1410	SHEET 7 OF 7
007	PROGRAMMER	DATE	IDENT 73 PANROLL

4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72
---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

000	IF NO-PREMIUM GO TO NO-PREM-RTN ELSE GO TO PREM1, PREM2
010	PREM3, DEPENDING ON SHIFT IN CODEIN, STOP 'INVALID PREMIUM
020	'CODE'.
030	
040	NO-PREM-RTN. MOVE ZERO-PREM TO PREMIUM-PAY. MOVE GROSS TO TOTAL-P
050	AY. GO TO OUTPUT-ROUTINE.
060	PREM1. COMPUTE SHIFT-PREM ROUNDED = GROSS * FIRST-SHIFT. GO TO
070	TOTAL-RTN.
080	PREM2. MULTIPLY GROSS BY SECOND-SHIFT GIVING SHIFT-PREM ROUNDED.
090	GO TO TOTAL-RTN.
100	PREM3. COMPUTE SHIFT-PREM ROUNDED = GROSS * THIRD-SHIFT.
110	
120	TOTAL-RTN. MOVE SHIFT-PREM TO PREMIUM-PAY. ADD SHIFT-PREM TO
130	GROSS GIVING TOTAL. MOVE TOTAL TO TOTAL-PAY.
140	OUTPUT-ROUTINE. MOVE RMK TO RM. WRITE EMPLOYEE-RECORD
150	GO TO NEXT-EMPLOYEE.
160	RWMD. CLOSE OUTPUT-PAY-FILE OPEN INPUT LIST-FILE.
170	NOTE PRINT OUT OF RECORD CREATED ON TAPE.
180	DISPLAY HEADING-RECORD UPON PRINTER1.
190	PRINT-RECORD. READ LIST-FILE AT END GO TO CLOSE-RTN.
200	IF CODEOUT IS EQUAL TO '999' GO TO CLOSE-RTN. DISPLAY
210	DISPLAY-RECORD1 UPON PRINTER1 GO TO PRINT-RECORD.
220	CLOSE-RTN. CLOSE LIST-FILE WITH LOCK. DISPLAY 'END OF EXECUTION'
230	STOP RUN.

# Index

\$3x Console Inquiry	9	Diagnostic Messages—Identification Division	35
1410/7010 COBOL Compiler Requirements	28	Diagnostic Messages—Procedure Division	36
ACCEPT Verb	20, 25	DIAGNOSTIC Operand	28
Acknowledgment	5	DISPLAY Verb	20
ADD CORRESPONDING Option	22	DIVIDE Verb	22
ADD Verb	21, 25	Editing Clause	26
Added Features		END DECLARATIVES	18
Data Division	16	End-of-File Switch (siu)	9
Procedure Division	24	ENDING-LABEL	14
ALL "literal"	26	ENDING Verb	24
ALPHABETIC	27	ENTER Verb	23, 25
ALPHANUMERIC	27	ENVIRONMENT DIVISION	8
ALTER Verb	23	Even Parity	11, 12
ALTERNATE AREAS	10	EXAMINE Verb	21, 27
APPLY Options	10	EXEQ Card Operand Options	28
Arithmetic Verbs	21	EXIT Verb	24
ASSIGN Clause	10	FD	13
Autocoder Subprograms	23	Figurative Constants	26
BEGINNING-LABEL	14	FILE-CONTROL Paragraph	9
BLANK WHEN ZERO Clause	16, 25	File Description Entry	13
Block Character-Count	13	FILE SECTION	13
BLOCK CONTAINS Option	13	Files and Records	11
Blocked Records		FILLER	15
Fixed-length	12, 14	Fixed-length,	
Variable-length	12, 14	unblocked records	12, 14
CALL Verb	23, 24	blocked records	13, 14
CARD-PUNCH xxx	10	FORTRAN Subprograms	23
Card Read Punch Records	12	General Information	25
CARD-READER xxx	10	GO TO Verb	23
Character Sets	26	Group Mark	16
Checkpoints	10	HIGH-VALUE	26
CLASS Clause	15, 25, 27	IBCOBOL Subprogram	29
Class Conditions	27	IDENT Field, Program-ID Card	28
CLOSE Verb	19	IDENTIFICATION DIVISION	6
COBOL Words, Listing	31	I-O-CONTROL Paragraph	10
COMMUNICATION-MODE	23, 24	I-O-SWITCH EOF-SIU	9
Communication Region (Resident Monitor)	8, 27	INPUT-OUTPUT SECTION	9
Compatibility Considerations	25	Input/Output Verbs	19
Compiler Directing Declaratives	18, 19	JUSTIFIED Clause	26
Compiler Directing Verbs	23	Key Words	6, 31
COMPUTE Verb	23	Label Processing	14, 18
Conditional Expressions	24	LABEL RECORD Clause	14
CONFIGURATION SECTION	8	Language Forms	6
Console Messages	36	For specific see individual clauses, etc.	
CONSOLE-PRINTER	8	Language Notations	6
CONSTANT SECTION	16	Level Indicator	13
Control Cards		Linkage Loader Control Cards	28
Monitor	28, 29, 30	LIST Operand	28
Linkage Loader	28, 29, 30	Literals	26
Control Card Requirements	29, 30	Load Mode	11, 25
CORRESPONDING Option		LOW-VALUE	26
ADD	21, 22, 25	Machine Requirements	5
SUBTRACT	22, 25	Mnemonic-Names	8
MOVE	20, 21, 25	Modes	
DATA DIVISION	11	Move	11
Data Manipulation Verbs	20	Load	11
DATA RECORD Clause	14	Even Parity	11
DECLARATIVES	18	Odd Parity	11
DEPENDING ON Option	13	Monitor Control Cards	28
Device-Names	8, 10	MONITOR-DATE	27
Diagnostic Messages—Data Division	35	MONITOR-SWITCH	9, 25
Diagnostic Messages—Environment Division	35	MOVE CORRESPONDING Option	20, 25
Diagnostic Messages—General	34	Move Mode	11

MOVE Verb	20	Rules for Arithmetic Verbs	21, 25
Multiple Subprogram COBOL Output	29	Sample Control Cards	
MULTIPLY Verb	22	Compile-and-Go	29
Non-Numeric Literals	26	Execution	30
NON-STANDARD	14	Sample Problem	36
Nonstandard Labels	14, 19	SECTIONS	
NOPCH Operand	28	CONFIGURATION	8, 10
NOTE Verb	24	INPUT-OUTPUT	9, 10
Numeric	27	FILE	13, 16
Numeric Literals	26	WORKING-STORAGE	16
OBJECT-COMPUTER Paragraph	8	CONSTANT	16
Object Time Error Analysis; Messages (Appendix D)	34	SELECT Clause	10
OCCURS Clause	15	Set A2—Character Set	26
Odd Parity	11	Set H2—Character Set	26
ON SIZE ERROR Option	22	SIGNED Clause	15, 25
OPEN-WITHOUT-REWIND	10	SIZE Clause	15, 25
OPEN Verb	19	SOURCE-COMPUTER Paragraph	8
Operand Options, EXEQ Card	28	Source Program Listing	34
Optional Words	6	SPACE	26
Organization of Source Program	32	Special-Names Paragraph	8
Padding	10	STANDARD	11, 13, 14
PADDING ON		Standard Tape Labels	11, 14, 19
APPLY	10	STOP Verb	24
Parity-Even, Odd	11, 13	Subprogram TITLE Card	28, 30
PERFORM Verb	23	SUBTRACT CORRESPONDING Option	22
PICTURE Clause	16, 25	SUBTRACT Verb	22, 25
PICTURE Symbols		Switches	8, 9
J, K, V, S, Z	16	Symbolic Units	10
POINT Clause	15, 25	SYSTEM-OUTPUT-PRINTER	8
PRINTER XXX	10	SYSTEM-OUTPUT-PUNCH	8
Printer Records	10	System Symbol MONITOR-DATE	27
Procedure Branching Verbs	23	System Units	8
PROCEDURE DIVISION	18	TALLY	27
PROGRAM-ID	6	Tape Files	9, 12, 25
Programming Techniques	25	TAPE-UNIT	10
Efficient Machine-Coding	24	Tape Units	10
Increase Compile Speed	25	TITLE Subprogram	24, 28
General	25	TITLE Card, Subprogram	28, 30
Qualification of Names	26	TRACE Operand	28, 29
QUOTE	26	Unblocked Records	
READ INTO Option	19, 25	Fixed-Length	12, 13, 14
READ Verb	19, 25	Variable-Length	12, 13, 14
Record Character-Count	12, 13, 14	USAGE Clause	15
RECORD CONTAINS Clause	13	Unit-Record Files	10, 12, 14, 25
Record Description Entry	14	UPON Option (DISPLAY)	20, 25
Record Formats		USE Verb	18, 19
Tape Files	12	USING Option	24
Unit-Record Files	12	VALUE Clause (Record Description entry)	16
Record Mark	16	VALUE Option (FD entry)	14
Recording Modes	11	Variable-length,	
RECORDING MODE Option	11, 13	unblocked records	12, 13, 14
REDEFINES Clause	15, 25	unblocked records with RCC	12, 13, 14
RENAMING Option	10	blocked	12, 13, 14
Requirements for Compilation	28, 29	W—Warning Message	34
Requirements for Execution	28	WORKING-STORAGE and CONSTANT SECTIONS	16
RERUN Option	10	WORKING-STORAGE SECTION	16
RESERVE Option	10	WRITE FROM Option	19, 20, 25
Resident Monitor's Communication Region Switch	8, 9	WRITE Verb	19, 20
Retention-Period	14	ZERO	26
ROUNDED Option	22		

Reader's Comments

IBM 1410/7010 Operating System (1410-PR-155)  
COBOL

Form C28-0327-1

From

Name \_\_\_\_\_

Address \_\_\_\_\_

Your comments regarding the completeness, clarity, and accuracy of this publication will help us improve future editions. Please check the appropriate items below, add your comments, and mail.

	YES	NO
Does this publication meet the needs of you and your staff?	_____	_____
Is this publication clearly written?	_____	_____
Is the material properly arranged?	_____	_____

If the answer to any of these questions is "NO," be sure to elaborate.

How can we improve this publication? \_\_\_\_\_ Please answer below.

- ☐ Suggested Addition (Page , Timing Chart, Drawing, Procedure, etc.)
- ☐ Suggested Deletion (Page )
- ☐ Error (Page )

COMMENTS:

No Postage Necessary if Mailed in U.S.A.

FOLD

FOLD

FIRST CLASS  
PERMIT NO. 81  
POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL  
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY  
IBM CORPORATION  
P.O. BOX 390  
POUGHKEEPSIE, N. Y.

ATTN : PROGRAMMING SYSTEMS PUBLICATIONS  
DEPARTMENT D9I

CUT ALONG LINE

FOLD

FOLD

**IBM**

International Business Machines Corporation  
Data Processing Division  
112 East Post Road, White Plains, N. Y. 10601